1994-06

Experimental hardware development for a pulsed ultrasonic data collection facility

Gatchell, Peter A.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

EXPERIMENTAL HARDWARE DEVELOPMENT FOR A PULSED ULTRASONIC DATA COLLECTION FACILITY

by

Peter A. Gatchell

June 1994

Thesis Advisor: John P. Powers

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1994	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE Experimental Hardware Development for a Pulsed Ultrasonic Data Collection Facility	5. FUNDING NUMBERS
6. AUTHOR(S) GATCHELL, Peter A.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE A

13. ABSTRACT *(maximum 200 words)* The work reported in this thesis used readily available components to implement a data acquisition system for a pulsed ultrasonic data collection facility. Use of hardware with controlling software is necessary to collect waveforms of acoustic potential at a given distance from the transmitting source. Precise movement and positioning of an acoustic receiver in the collection plane is accomplished by use of an xy coordinate motor-driven slide assembly. A signal generator and transient digitizer transmit and digitize the signal, respectively. These components are brought together synchronously using LabVIEW instrumentation software. This work provides an efficient means to collect waveform data which can be used to verify computer code written previously for the purpose of modeling pulsed ultrasonic acoustic diffraction patterns.

14. SUBJECT TERMS LabVIEW, C, diffraction, ultrasonic propagation			15. NUMBER OF PAGES 66
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

EXPERIMENTAL HARDWARE DEVELOPMENT FOR A PULSED ULTRASONIC DATA COLLECTION FACILITY

by

Peter A. Gatchell
Lieutenant, United States Naval Reserve
B.S., Texas A&M University, 1985

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1994

Author: [redacted]
Peter A. Gatchell

Approved by: [redacted]
John P. Powers, Thesis Advisor

[redacted]
Ron J. Pieper, Thesis Second Reader

[redacted]
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

The work reported in this thesis used readily available components to implement a data acquisition system for a pulsed ultrasonic data collection facility. Use of hardware with controlling software is necessary to collect waveforms of acoustic potential at a given distance from the transmitting source. Precise movement and positioning of an acoustic receiver in the collection plane is accomplished by use of an xy coordinate motor-driven slide assembly. A signal generator and transient digitizer transmit and digitize the signal, respectively. These components are brought together synchronously using LabVIEW instrumentation software. This work provides an efficient means to collect waveform data which can be used to verify computer code written previously for the purpose of modeling pulsed ultrasonic acoustic diffraction patterns.

TABLE OF CONTENTS

I. INTRODUCTION

The angular spectrum approach is a widely used tool to explore the propagation characteristics of continuous ultrasonic sources [Ref. 1]. Associated mathematics for this technique are well understood and are useful in analyzing and predicting acoustic diffraction patterns emitted by ultrasonic and optical sources [Ref. 2]. Many practical applications use pulsed sound instead of continuous sound for acoustic imaging, tissue characterization, and other specialized functions such as mine detection. Computer programs have been developed for prediction of the diffraction patterns from pulsed sources, allowing time-efficient reconstruction and simulation of patterns [Ref. 3]. However, this simulated data must be verified with experimental data. The purpose of this thesis was to construct and synthesize a pulsed ultrasonic collection facility, using readily available components, which could collect these waveform diffraction patterns.

Using the measured waveform from the collection facility, diffraction patterns can be reconstructed and compared with those simulated on computer. The collection facility is needed in order to confirm that previously written computer code is valid and reasonable and that the simulated diffraction patterns are accurate. The collection facility described here enables numerous sets of waveform data to be collected using various combinations of transmitters and receivers. Operator controls are at a minimum so data collection can be implemented in a short period of time. The receiving plane utilized in the program can also be resized to accommodate user-defined requirements.

Chapter II of this thesis consists of a problem description, including both hardware and peripheral software components used to implement the system. Chapter III provides program explanations of COLLECT.VI code and how it was used to implement the collection facility system. System operation is described in Chapter IV and ends with a brief discussion on waveform data format. Following the summary of Chapter V, Appendix A gives the C programming language source code used to drive the stepper motors. Appendix B provides the LabVIEW graphical source code for the collection facility program COLLECT.VI.

II. PROBLEM DESCRIPTION

A general ultrasonic data collection facility consists of a water tank, scanning device, pulse generator, waveform digitizer, and, for a central point of control for all these devices, a computer. (See Figure 1.) The water tank is the facility which confines the acoustic medium. The motor driven scanning device, transmitter, receiver, and associated mounting hardware are used for acoustic wave collection within the facility. A pulse generator is needed to drive the transmitter and a digitizer is used for analog-to-digital conversion of the data. The digitizer puts the signal in a format which can be easily recorded to disk on a Bernoulli drive. A computer enables synchronization of all these components.

For assembly, necessary components had to be identified. The water tank, xy coordinate motor driven slides, and acoustic receiver were available components around which the system was built. Other components, such as the scanning tank frame and plexiglass pieces, were fabricated to nominal system specifications. These specifications were inferred from other scanning tank designs and integrated with facilities available in the laboratory [Ref. 4]. Other equipment, such as pulse generator and transient digitizer, were recognized as essential pieces because their use could be controlled via a General Purpose Interface Bus (GPIB). A personnel computer (PC), using an Intel 486DX 33 MHz processor, served as the central point of control for all other hardware. LabVIEW, a graphical program for instrumentation control, linked the different instrumentation

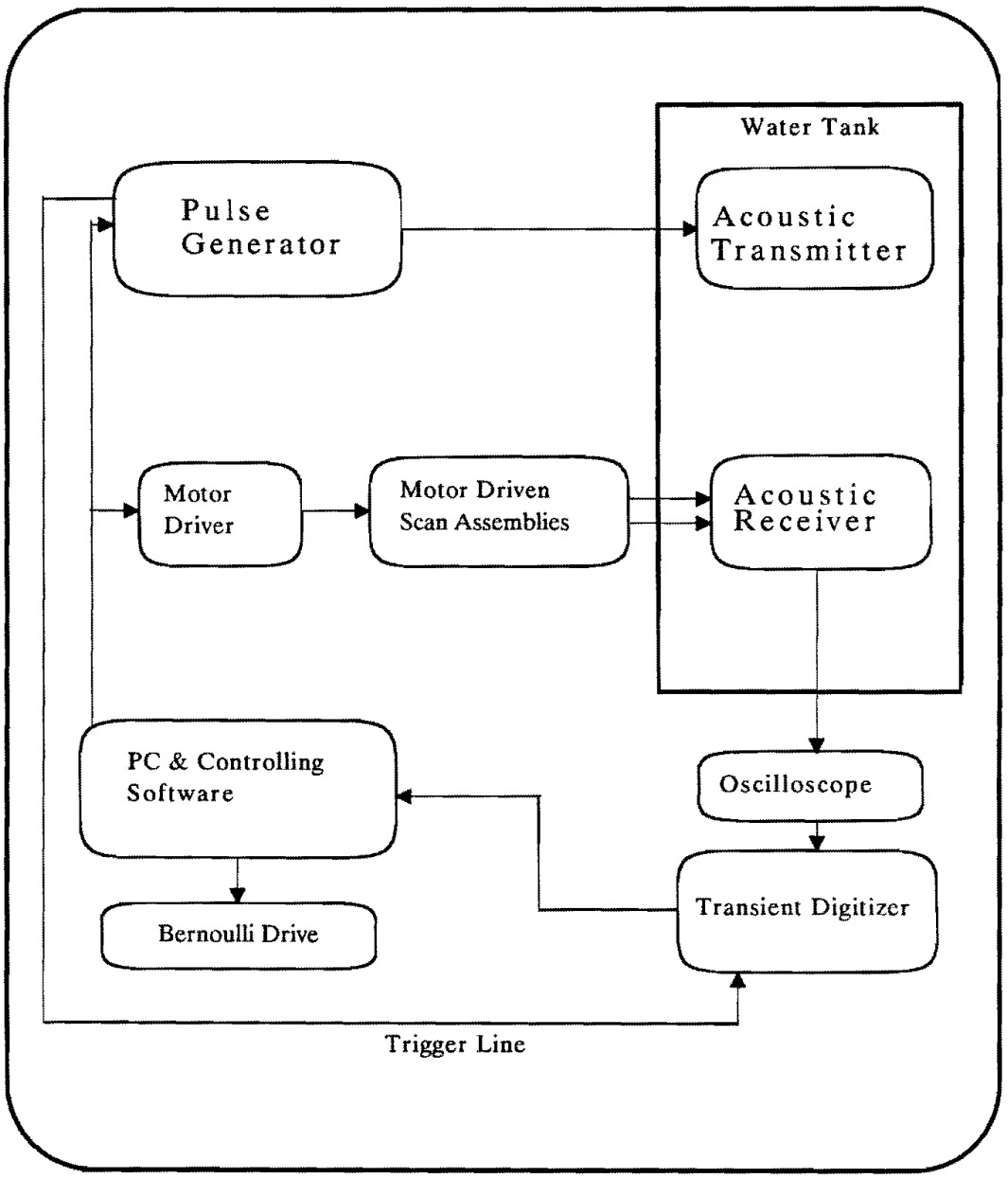

Figure 1. Overall Experimental Arrangement

and motor drivers. The following two sections outline implementation of both hardware and peripheral software in the collection facility design. A description of each component is given along with how it is used.

A. IMPLEMENTATION OF HARDWARE

1. Scanning Tank and Frame

A scanning tank with outside dimensions 76.4 by 33.6 by 34.5 centimeters was used to contain the water. Constructed of aluminum sheet metal and wood, it was sealed with silicon rubber and lined on the inside with styrofoam. The styrofoam served to attenuate acoustic echoes. The tank was filled with tap water which had been filtered using an ion exchanger. This was done to remove impurities and provide for a homogeneous medium through which the acoustic waveform could travel. The tank was placed on a cart that could be wheeled in and out of the tank frame, in addition to allowing insertion and removal of water.

The tank frame was constructed of welded 0.6 centimeter aluminum bar stock in a horizontal rectangular box frame design so that it could be mounted upon and between two laboratory tables. XY motor-driven slides for vertical and horizontal positioning of the acoustic receiver were mounted at one end of the frame. A movable slide was also constructed for attachment of the acoustic transmitter. This slide was positioned between the box rails and permitted arbitrary positioning of the transmitter within the water tank itself. Separate plexiglass mounts were fabricated and used to position both the acoustic transmitter and receiver in the water. The acoustic transmitter mount was attached to the

hand-adjustable slide and the receiver mount was attached to the y (vertical) motor-driven slide.

2. Stepper Motors, Motor Driven-Slide, and Motor Driver

Two stepper motors, in conjunction with motor-driven slides (xy coordinate motor-driven slides), move the acoustic receiver along vertical and horizontal axes. For this application, the motors use a "single step mode" which means that for each programmed step, the shaft will turn 1.8 degrees. As a reference, 5100 steps will move a slide 2.5 centimeters (1 inch). [Ref. 5]

The xy coordinate motor-driven slide incrementally positions the acoustic receiver during data acquisition. Each motor drives a screw upon which a slide is attached. The screws are approximately 53 centimeters in length and contain eleven threads per centimeter. One screw moves the acoustic receiver along the vertical direction (y-axis), while the other moves the vertical slide/acoustic receiver along the horizontal direction (x-axis). Thus, precise positioning of the acoustic receiver is accomplished by stepping (incremental rotation of the shaft) the motors. [Ref. 5,6]

Current and voltage supplied to the motors comes from a dual stepper motor driver. This motor driver is controlled via programming language C code during data collection, or by a separate Disk Operating System (DOS) program for manual operation. The motor driver is basically a computer-controlled power supply and is connected to the PC using a standard parallel printer cable. [Ref. 6]

3. Acoustic Transmitter and Receiver

Two acoustic transmitters operating at 2.25 MHz were used and could be interchanged. The first one contained a rectangular shaped active transmitting area that measured 5.1 by 0.65 centimeters while the second contained a circular active transmitting area with a diameter of 2.55 centimeters. Both transmitters, used interchangeably, were connected directly to the pulse generator using coaxial cable and pulsed with one cycle of a 10 volt amplitude sine wave.

A piezoelectric acoustic receiver with an active receiving diameter of 0.2 centimeters across was used to detect the pressure wave in the water. Coaxial cable connected the receiver directly to an oscilloscope.

4. Pulse Generator

A Wavetek Model 270 12 MHz Programmable Function Generator provided the pulsed sine wave to be transmitted. Appropriate values for frequency (2.25 MHz), amplitude (10 volts), signal offset (0), signal mode (burst), function type (sine wave), and burst mode (one cycle) were programmed using GPIB procedures. Although manual entries could be made, the values were incorporated into the LabVIEW program to allow automatic function generator initialization. The pulse generator was connected directly to the PC using GPIB cable and to the transmitter using coaxial cable. An additional coaxial cable was connected from the "synchronous output" of the pulse generator to the external "trigger in" of the digitizer. This trigger was used to initiate the data acquisition process of the digitizer. [Ref. 7]

5. Transient Digitizer

Upon reception, the received waveform was digitized and transferred to a file. This was done using Tektronix RTD720A Transient Digitizer. The RTD720A is a high bandwidth, fast sample rate, long record-length digitizer designed to accurately capture fast transient events. A display provides viewing of each received waveform burst. Sampling of each waveform burst is accomplished well above the required Nyquist frequency and is digitized to a record length of 512 bytes. This digitized record of the waveform is what is actually transferred to a file on the computer. Coaxial cable was used to make a connection between "50 ohm channel 1 input" of the digitizer and from the rear "channel 1 output" of the oscilloscope. An additional connection was made for the trigger as mentioned previously. The digitizer is automatically initialized using GPIB procedures within the LabVIEW program such that the received waveform is captured with all necessary details. [Refs. 8 and 9]

6. Oscilloscope

As can be seen from Figure 1, a 100 MHz oscilloscope was used between the acoustic receiver and transient digitizer. The oscilloscope was used in part as a tool for manual waveform analysis. An additional feature was that it provided a 50 ohm impedance output to match the digitizer input impedance. Coaxial connections were made directly from the acoustic receiver to the front panel "channel 1 input" of the oscilloscope and from the rear "channel 1 output" to the "50 ohm channel 1 input" of the transient digitizer. [Ref. 10]

7. Personal Computer (PC)/Bernoulli Drive

An IBM compatible PC with an Intel 486DX 33 MHz processor and 8 megabytes (MB) of random access memory (RAM) provided a suitable platform from which all software applications could be run. The fast processor speed makes for easy system programming/debugging. All applicable software for the collection facility was loaded onto the internal 105 MB hard drive.

For waveform data storage, an Iomega Bernoulli Box II 20 MB 5.25 inch removable dual external drive was used. A single 5.25 inch drive was sufficient since it satisfied the requirement of ample storage space and removability. Data could then be removed and transferred to another PC for processing. Writing to a file on the Bernoulli drive is simply a matter of specifying the correct path. An adapter board and cable were provided with the Bernoulli Box II for integration into the PC system. [Ref. 11]

8. National Instruments General Purpose Interface Bus (GPIB)

The GPIB is an interface system through which interconnected electronic devices communicate. In this case, an IEEE 488.2 industry standard AT-GPIB board controlled the pulse generator and transient digitizer. Incorporation of this device involved installing the AT-GPIB board onto the PC motherboard and connecting the cables to the pulse generator and transient digitizer. Commands embedded in the LabVIEW program activated these devices synchronously for system operation. [Ref. 12]

B. IMPLEMENTATION OF PERIPHERAL SOFTWARE

1. National Instruments (NI) 488.2 Software Package

The NI-488.2 software was installed on the PC's internal hard drive under the directory "AT-GPIB". The CONFIG.SYS file was modified so that the AT-GPIB driver was loaded each time the PC was started. Hardware diagnostic utilities, included with the software, were run to insure correct AT-GPIB board installation. No other files included with this software package were used. All necessary peripheral information was entered into a similar program contained within the LabVIEW environment. [Ref. 12]

2. ARRICK Robotics Software Package

Two programs within this software package were used to drive the stepper motors. All programs were installed on the PC's internal hard drive under the directory "MD-2". Manual operation of the motors was effected in the DOS environment using keyboard commands. Once in the MD-2 directory, the command "MD-2MOVE", typed at the DOS prompt, provided manual operation. An interactive motion control program, MD-2MOVE, displays all motion control functions in a simplified environment. Initial positioning of the acoustic receiver was done by using this program with motion control function settings as shown in Table I. Once the motors were moved to their initial position, this program was exited back to the DOS environment. The robotics software package also included a library of subroutines written in C programming language called MD-2SUBC.C. These subroutines contained a collection of motor control functions which were slightly modified for use in the LabVIEW data collection program. Several lines of

code were written and merged with these subroutines to allow control of motor parameters and functions. (See Appendix A for the revised code.) The code was compiled and linked to LabVIEW using WATCOM C/C++. Motor control could then be performed within the LabVIEW environment. [Ref. 5]

3. WATCOM C/C++ 32-Bit Compiler

To use the MD-2SUBC.C subroutines, it was necessary to compile them first using WATCOM. This brand of 32-bit C compiler is the only type supported by

TABLE I. MOTION CONTROL FUNCTION SETTINGS

Function	Setting
Motor	3,4
Move Control	F5, F6, F7, F8
Step Type	Single
Backlash	0
Step Increment	5100
Speed	7000
Switch Status	No
Current Position	0
Key Check	Yes
Switch Check	No
Power Down	Yes

LabVIEW. The compiler was loaded on the PC's internal hard drive in the DOS environment under the directory WATCOM. Code modifications were made to MD-2SUBC.C, saved to a different name (PREPOS.C or MOVE.C), and compiled/linked in the C:>WATCOM\BIN subdirectory using the commands as shown:

C:>WATCOM\BIN WMAKE /F PREPOS.LVM

or

C:>WATCOM\BIN WMAKE/F MOVE.LVM.

The files PREPOS.LVM and MOVE.LVM link PREPOS.C and MOVE.C into the LabVIEW operating environment. All modification and debugging of PREPOS.C and MOVE.C was accomplished in C:>WATCOM\BIN subdirectory; Appendix A shows the actual code modifications. Each corresponding block of code was merged into MD-2SUBC.C and saved under the appropriate file name. [Refs. 5 and 13]

The hardware and peripheral software are essential components in the data collection system. However, for full data collection, it is necessary for all components to operate synchronously. LabVIEW provides the platform from which this can be achieved.

III. LABVIEW INSTRUMENTATION SOFTWARE

A. OVERVIEW

In this chapter, all hardware and peripheral software are tied together using LabVIEW graphical programming techniques. An introduction to LabVIEW is followed with program sections as they occur in the graphical code, COLLECT.VI, that controls the data acquisition facility.

1. Introduction

Much like C or BASIC, LabVIEW is a program development application. However, these other programming systems use text-based languages to create lines of code while LabVIEW uses a graphical programming language to create programs in block diagram form. LabVIEW is a general-purpose programming system with extensive libraries of functions and subroutines which are identical in operation to those in conventional language programs. The programs developed in LabVIEW are called *virtual instruments (VIs)* because their appearance and operation imitate actual instruments. VIs have an interactive user interface called a *front panel* and a source code equivalent called a *block diagram.* The front panel simulates the panel of a physical instrument which contain controls and indicators. Data input is accomplished by using a mouse and keyboard. Instructions are received from the block diagram which is actually a pictorial solution to the programming problem. [Ref. 14]

The program designed for the data collection system is called COLLECT.VI. Upon aligning the receiver with the centerline axis of the transmitter face using MD-2MOVE, running COLLECT.VI is all that is needed for data collection. Any changes made to the receiver collection plane dimensions or instrumentation setup are performed within COLLECT.VI.

2. Data Collection Program Algorithm

Figure 2 shows the general processes that COLLECT.VI steps through. The user sets the program variables in the case selection block. The variables are determined by the choice of receiving plane size made on the front panel of the program. Features, such as options to Initialize Wavetek 270, Tektronix RTD 720A, and Prepositioning of Receiver, were also built in for adjusting the setup. The core of the program, called the Collection Chamber Loop, accomplishes the data collection. This loop runs continuously until completion of data collection, at which time the program stops.

B. DATA COLLECTION PROGRAM (COLLECT.VI)

The figures in Appendix B (Figures 3-7 and 9-28) are LabVIEW graphical code block diagrams. Each block diagram depicts operations which are performed at that point in the program. In every block diagram of Figures 3-7 and 9-28, note that operations are encompassed by a *frame* or *case* structure, and that every *frame* or *case* structure contains either a numeral or a "True" or "False" label at the top. Some structures contain

other structures, such that all inner structures will complete first, prior to execution of the outer structure. For example, Figure 7 contains a frame structure with the numeral "0" at

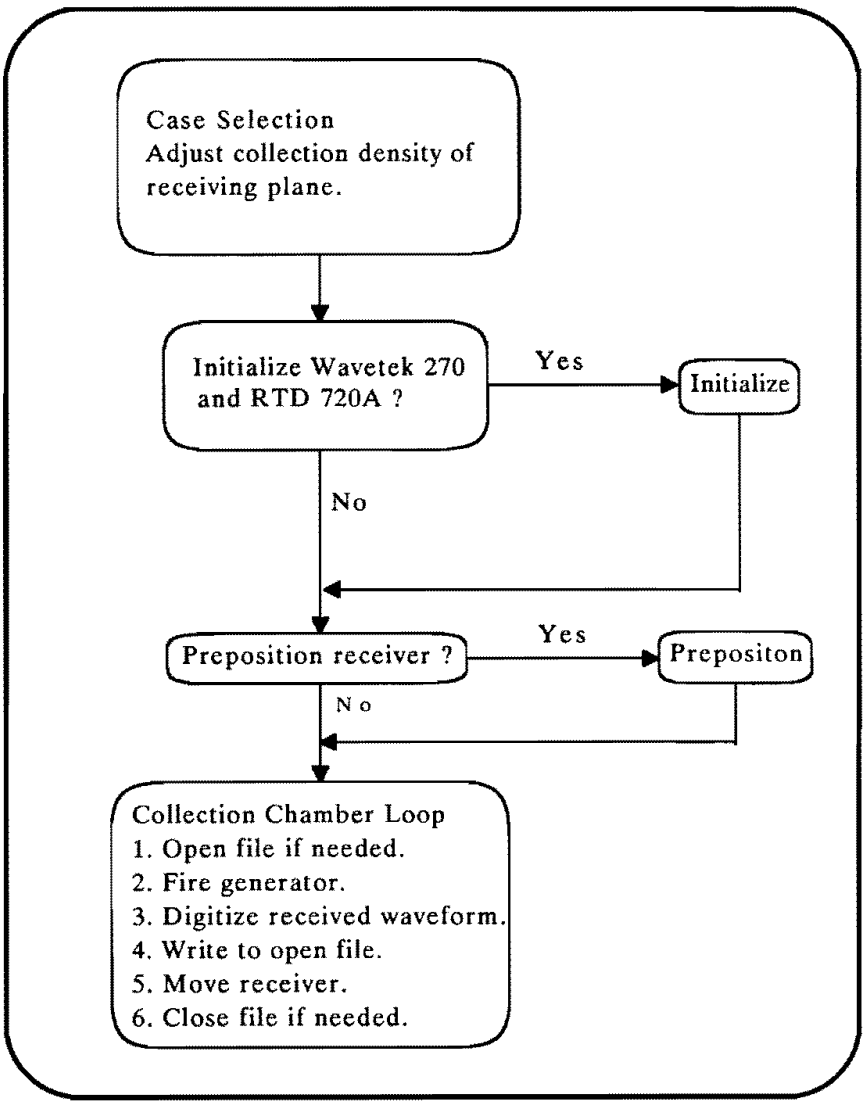

Figure 2. COLLECT.VI Block Diagram Algorithm

the top and a case structure located within that frame with a numeral "3" at the top. Note that the frame structure has a border that replicates photographic film, thus the name *frame*. The case structure can have a numeral or a "True" or "False" label at the top while the frame only uses a numeral. In addition, the case structure uses a different style of border as can be seen in the figure. Another example, Figure 24, contains layered frame and case structures. The outer frame "4" contains two loops. The outside loop contains the inside loop which also contains a frame labeled "7" at the top. Within this frame, there is a case structure labeled "True" and a frame structure labeled "2". The structures are sequential, such that operations within each structure are not begun until the previous structure's operations have completed. All block diagrams are annotated for easy interpretation and are titled "Block Diagram xxxx". The "xxxx" are a combination of numerals and or letters identifying the structure and to connote the point that it occurs in the program sequence. Each figure contains numerous labels to help explain block diagram icon operations. Further discussion of block diagram icons can be found in the LabVIEW User Manual [Ref. 14].

1. Front Panel

Operating controls (shown in Figure 3) provide for the initialization of instruments, prepositioning of receiver and for adjusting the receiving aperture size. The operator has a choice of collection-plane densities: 32x32, 64x64, and 128x128, for actual data collection, or a 4x4 for testing purposes. Since the receiving aperture is actually fixed

at four inches by four inches, choosing a different density will change the number of collection points within the aperture. Indicators are shown at the bottom right to keep track of receiver position.

2. Case Selection

Selection of program variables are shown in Figures 4, 5, 6, and 7. The operator chooses a receiving plane density and, from that choice, the program variables *total points*, *row/column length*, and *steps to send* are set. Figures 4, 5, 6, and 7 perform the same functions in regards to calculation and storage of values. For example, if the operator chooses a 32x32 receiving plane density, the program will execute case 1 (see Figure 5). *Total points* is the total number of data collection points in the receiving plane and is computed by multiplying 32 by 32 as shown. This value is displayed on the front panel. *Row/column length* is used in the loop structures of Figures 15 through 28 for keeping track of receiver position. For a 32x32 collection density, *row/column length* is 32. *Steps to send* are precalculated numbers used in MOVE.C. These values are the number of steps between each data collection point. Shown in Figure 8 is an example of the 4x4 collection density plane. This model was extended to a 32x32, 64x64, and 128x128 collection density plane for proof of concept and calculation purposes. Calculations for a 4x4, 32x32, 64x64, and 128x128 collection density planes and the number of steps needed for MOVE.C are shown in Table II. The number of steps per inch was determined to be 5100. Since the number of inches

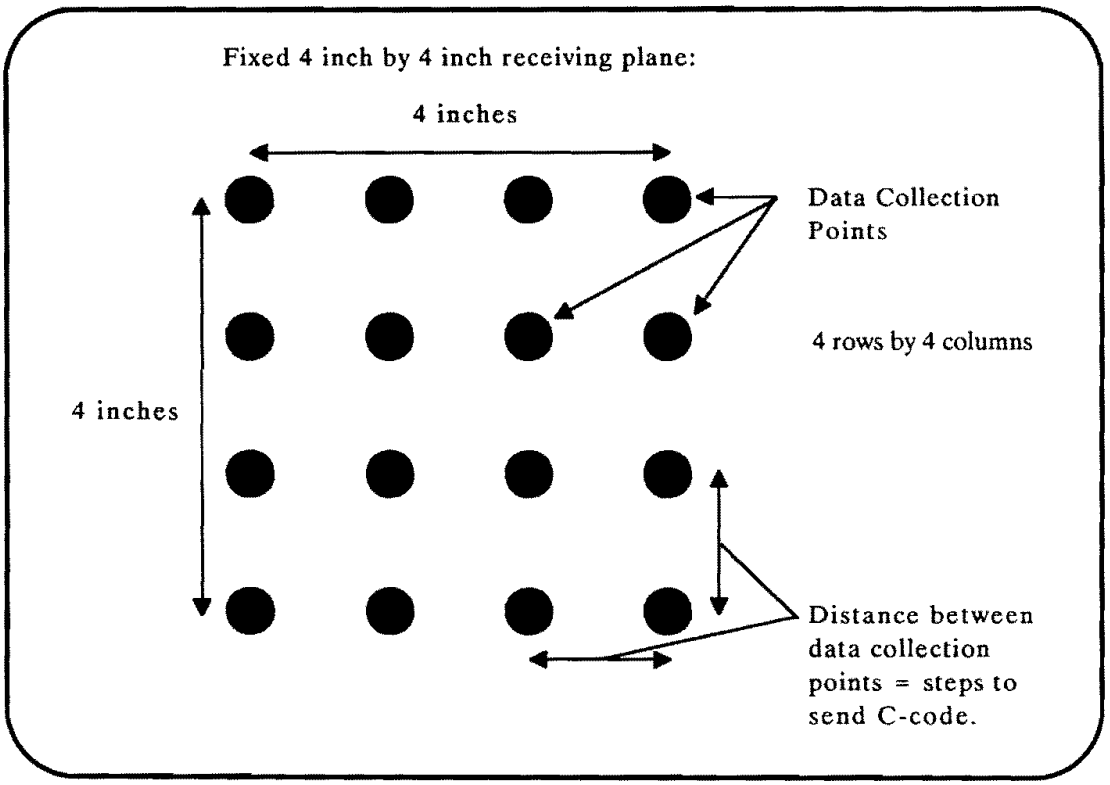

Figure 8. 4x4 Collection Density Plane

TABLE II. 4x4 COLLECTION PLANE DENSITY CALCULATIONS

Collection Density	Steps to Send
4x4	20400 / 3 = 6800
32x32	20400 / 31 = 658
64x64	20400 / 63 = 324
128x128	20400 / 127 = 160

per side for a fixed 4 inch by 4 inch receiving plane is four, the total number of steps per side is 4 * 5100 = 20400. Each side contains N = 4, 32, 64, or 128 collection points per side. Thus the number of steps between each collection point is 20400/(N-1) which is the value of *steps to send* as shown in Figures 4, 5, 6, and 7. For prepositioning, the transmitting and receiving axes are collinear and centered within the receiving plane. Thus, the number of steps necessary to start data collection with the receiver positioned at the top left corner is: steps for prepositioning = 1 / 2 * 20400 = 10200 (see Figure 13). Note that all numbers in Table II are rounded to an even integer. This is because MOVE.C and PREPOS.C will only accept even-valued integers.

3. Initialization of Equipment

If the operator chooses *yes* on the front panel for *Initialize Instruments*, then preset values will be transmitted to Wavetek 270 and Tektronix RTD 720A, as shown in the TRUE case structure of Figures 9 and 10. Shown under *settings* within the border are the preset values transmitted. The operating tool is used to change these values if needed. The LabVIEW icon labeled *SEND* transmits data bytes (preset values) to the

corresponding instrument indicated by *GPIB address*. *Mode* indicates that a "End Or Identity" message will be sent at the termination of the data bytes. If *no* is chosen on the front panel, all initialization is bypassed and no operations will be carried out as shown in the FALSE case structure of Figure 11. At this point, manual entries would have to be made to the Wavetek 270 and Tektronix RTD 720A prior to running the data collection program.

4. Wait Time

As shown in Figure 12 and as interspersed throughout the program, various structures contain "wait" functions. These functions, whose icon looks like a timer, will cause the program to wait a specified number of seconds (or fraction thereof), prior to moving on to the next structure. Thus COLLECT.VI will remain idle upon executing this frame. The purpose is to allow completion of previous program tasks and synchronization of operations.

5. Prepositioning of Receiver

The operator has the discretion to preposition the receiver from the front panel. Figure 13 executes if *yes* is selected on the front panel. Shown within case 0 of Figure 13 are *Steps for Prepositioning* and *Code Interface Node*. *Steps for prepositioning* is the number of steps sent to PREPOS.C via the *Code Interface Node* to activate the stepper motors and position the receiver. As mentioned previously in Chapter II, (under WATCOM C/C++ 32-Bit Compiler), PREPOS.C and MOVE.C are linked into the LabVIEW environment. The *Code Interface Node* is the tool by which LabVIEW VI's are

able to link with external source code written in conventional programming languages. The *Code Interface Node* of Figure 13 is a block diagram node associated with the C programming source code PREPOS.C. Other nodes used in COLLECT.VI (Figures 23, 25, and 27) are associated with MOVE.C. Each of the nodes used in COLLECT.VI send information to the external code for processing. If *no* is selected, case 1 of Figure 14 will execute. At this point COLLECT.VI will remain idle, prepositioning will not occur, and the program will begin data collection at the current location.

6. Data Collection Loop

Two loop structures were used to perform the repetitive processes for data collection. Outer frame structure 4, as is common to Figures 15 through 28, contain an outer loop and an inner loop. Each loop contains a "**N**" on the upper left corner. This connotes the total number of times the loop will repeat and is always equal to *row/column length*. Thus, each loop will execute for **N** = 4, 32, 64, or 128 times. The inner loop will complete **N** cycles for each of the **N** rows of the outer loop. When the outer loop reaches the value of **N**, and the inner loop completes data collection over **N** columns, the program stops and data collection is complete. An "**i**" located on the lower left corner of each loop records the number of times the loop has executed, starting with the numeral 0.

The outer loop keeps track of the receiver row position and makes the determination of opening a new file. (A new file is opened for each row.) Outer loop functions are shown to the right of the inner loop in Figures 15 through 28. Every time the outer loop executes, a new file is opened. (The previously opened file is closed by the

inner loop.) To enumerate each row correctly, "**i** + 1" is calculated and appended to the file name *data.txt*, (e.g., *data1.txt*, *data2.txt*, *data3.txt*,...etc.). Thus a new file name is created for each row. This new file name is then appended to a path containing the desired directory for data file storage. COLLECT.VI will store all files in the directory name *dat32a* on the external Bernoulli drive *d.* The new file is then opened for writing.

Inner loop structures, which contain inner frames 0 through 8 (common to Figures 15 through 28), compromise the program core. Prior to digitizing, the Tektronix 720A is armed just before firing the pulse generator (Figures 15, 16, and 17). Arming commands are sent to Tektronix 720A located at GPIB address 20 (Figure 15), a wait time is inserted for synchronization (Figure 16), and fire commands are sent to Wavetek 270 located at GPIB address 4 (Figure 17). After firing, the waveform is transmitted through the water and received by the Tektronix RTD720A. Additional time is inserted to allow for digitization of the waveform (Figure 18). In Figure 19, the *RECEIVE* icon reads up to 2000 waveform data bytes from GPIB address 20. Next, COLLECT.VI writes this information to the open file (Figure 20). A Wait time of one second was programmed in after this operation (Figure 21).

To determine the current receiver position in the collection plane, the inner loop uses *row/column length* and the "**i**" counter from the outer loop. These two values are multiplied together, added to "**i** + 1" form the inner loop, and compared to the computation of "**i** + 1" from the outer loop times *row/column length*. If the comparison is equal, the TRUE case structure (Figure 22) within frame 7 of the inner loop structure will

execute. COLLECT.VI has determined that the receiver just collected data from the last column of the present row and the current file is closed. The next frame within this case structure (Figure 23) contains a code interface node to activate the vertical stepper motor. The proper number of steps to move and direction for each stepper motor is sent to MOVE.C such that the receiver is moved down to the next row. After a one-half second wait (Figure 24), the receiver is moved back to the beginning of the row (Figure 25). This is accomplished using an additional code interface node for MOVE.C. Note that the correct number of steps to move the horizontal stepper motor has to be calculated within this frame structure. Wait time is again employed at the end of this operation (Figure 26).

Figure 27 shows a FALSE case structure which is executed when the conditions for a TRUE case structure of Figure 22 are not met (i.e., the receiver is not positioned at the end of the present row). In this case, the code interface node activates the horizontal stepper motor via MOVE.C and the receiver is moved to the next data collection point of that row. COLLECT.VI is idle for two seconds (Figure 28) at the completion of receiver movement to allow for any vibrational movements of the receiver plexiglass structure to cease.

IV. SYSTEM OPERATION

A. GENERAL PROCEDURE

Steps have been developed to prepare the data collection system for use. These steps are general guidelines which have evolved from use of the system.

1. Hardware Configuration

After the scanning tank has been filled and sited under the frame, both transmitter and receiver can be immersed and positioned relative to each other. It is necessary that the transmitter be firmly mounted onto the plexiglass fixture using the rubber gasket such that a watertight seal is made between the transmitter itself and coaxial cable. The movable slide upon which the plexiglass and transmitter are attached is then positioned at the desired distance from the receiver. This distance is from the face of the transmitter to the face of the receiver. Both transmitting and receiving surfaces must be parallel to each other. It is best to manually operate the Wavetek 270 pulse generator and Tektronix RTD720A at this point. By manually pulsing the generator, captured waveform bursts can be observed on the Tektronix RTD720A and any necessary adjustments recorded for later incorporation into COLLECT.VI. As a final check, insure that all cables are connected to the various instruments and that there are no obstructions impeding the movement of vertical and horizontal slides.

2. LabVIEW Front Panel and Diagram

Choose the desired options on the front panel. If it is necessary to change the settings of Wavetek 270 and Tektronix RTD720A (Figures 9 and 10), go into the block diagram and use the LabVIEW operating tool to enter new values. Note that all changes will be effected using the LabVIEW operating tool. Other changes which might be necessary are *steps to send C-code* (Figures 4, 5, 6, and 7) and *Steps for Prepositioning* (Figure 13). If the receiving plane is changed from a 4 inch by 4 inch plane to a 3 inch by 3 inch plane, then *steps to send C-code* and *Steps for Prepositioning* will have to be recalculated following the same guidelines as shown in Figure 8. The directory and file name to which the data is written can be changed too. This is done in Figure 15. To the far right of the outer loop structure contained within frame 4 are the file directory and name. As programmed, COLLECT.VI writes to drive *d* into directory *dat32a* as shown by *d:\dat32a*. It is necessary that this directory be created prior to running COLLECT.VI. COLLECT.VI will create all necessary files into which data is stored, but it will not automatically create the directory to which the files are written. The name of the file is shown by *data%d.txt*. The name *data* and the three letter extension *.txt*, can be changed, but the characters *%d* must be left unaltered in the corresponding position as shown. The characters *%d* are used by LabVIEW to insert an integer which corresponds to the current row of the collection plane.

3. Waveform Data and Format

After COLLECT.VI is run, the waveform data can be viewed upon exiting LabVIEW and going into the file itself. Each individual digitized waveform is preceded by the word "curve" as seen in the file. Actual waveform data is written in binary format. All recorded data is ready for immediate processing after the program is finished. An acoustic waveform collected from the system is shown in Figure 29. Similar to what is seen on the Tektronix RTD 720A, the waveform is plotted in relative amplitude vs. time. This waveform was taken on row 32, column 32 of a 64x64 collection density setup using a circular acoustic transmitter. When the Tektronix 720A digitizes this waveform, it breaks the time axis into 1024 points. (Note that the Tektronix RTD720A, as described in this thesis, was programmed for 512 points as shown by *acquisition length* in Figure 10.) Figure 30 is the actual ultrasonic diffraction pattern. This graph was created by taking the relative amplitude of the 80^{th} point for each of the 4096 individual data collection points of the 64x64 collection density plane. The shape of the source (transmitter), as can be inferred from Figure 30, is circular.

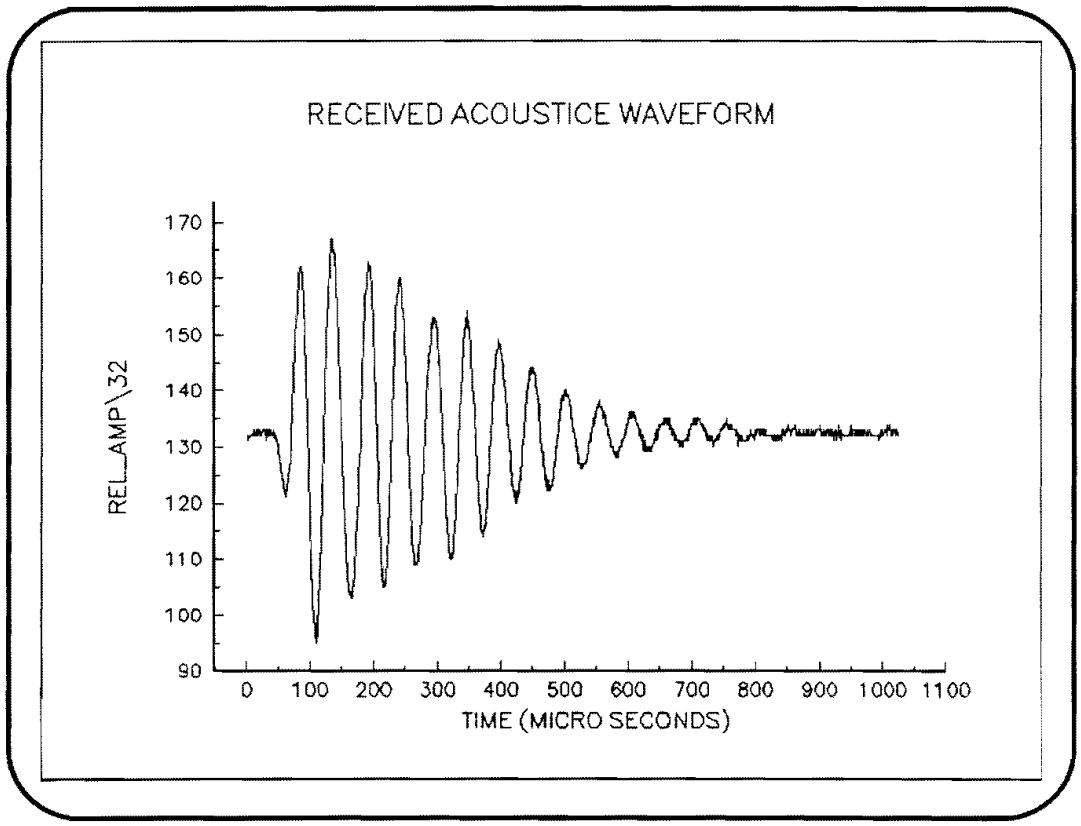

Figure 29. Waveform Pulse from a 64x64 Collection Density.

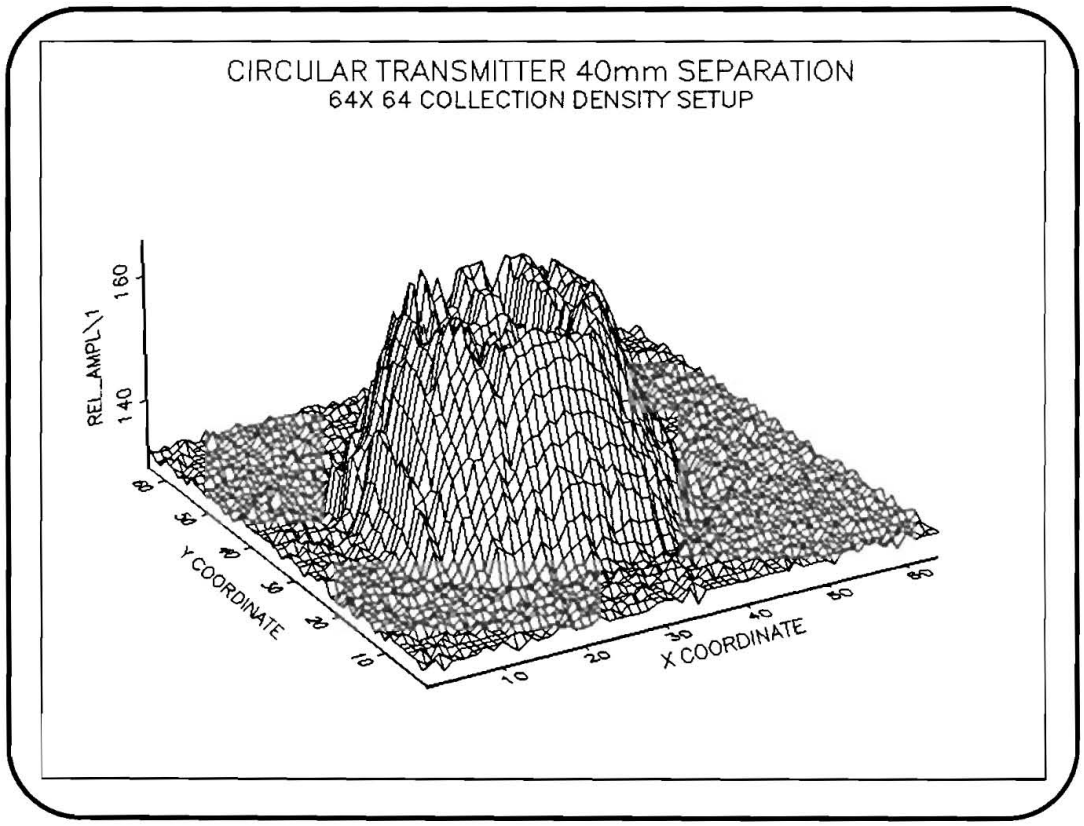

Figure 30. Ultrasonic Diffraction Pattern from a 64x64 Collection Density.

V. SUMMARY

This thesis presented an effective method by which a pulsed ultrasonic waveform collection chamber system could be constructed. Most components used were those which were readily available in the laboratory or could be fabricated at minimal cost.

Initially, the primary focus was on identifying candidate hardware and software for the system. It was decided that all components would be chosen in view of how they could be implemented with LabVIEW. This was done so that data collection would be totally automated, requiring no attention from the operator while COLLECT.VI was running. It has been determined that several modifications to the system could enhance the data collection process. One such improvement would be the ability to control the stepper motors manually while in the LabVIEW environment instead of from the DOS environment using MD-2MOVE. These improvements would ease positioning the receiver. Other modifications could have been made to COLLECT.VI in regards to directory/file manipulation, such as enabling changes to be made from the front panel.

The end goal of this thesis effort was to build a system which would be simple and effective. These objectives have been accomplished using available materials as outlined.

APPENDIX A. SOURCE CODE FOR PREPOS.C AND MOVE.C

The following source code is that which was used to modify MD-2SUBC.C. This code was inserted within the body of MD-2SUBC.C and saved under PREPOS.C or MOVE.C.

PREPOS.C:

```
CIN MgErr CINRun(int32 *var1)
{
    motor = 34;                 /* initialize ports for motors 3&4 */
    md_2init( );
    move_action = 's';
    steps_to_move[3] = *var1;
    steps_to_move[4] = 0;
    speed[3] = 14000;
    direction[3] = 'f';
    md2_move( );
    steps_to_move[3] = 0;
    steps_to_move[4] = *var1;
    speed[4] = 14000;
    direction[4] = 'f';
    md2_move( );
    md_2reset( );
    return noErr;
}

CIN MgErr CINInit(void) {return noErr;}
CIN MgErr CINDispose(void) {return noErr;}
CIN MgErr CINAbort(void) {return noErr;}
CIN MgErr CINLoad(RsrcFile rf) {return noErr;}
CIN MgErr CINUnload(void) {return noErr;}
CIN MgErr CINSave(RsrcFile rf) {return noErr;}
```

MOVE.C:

```
char            dir3;
char            dir4;

CIN MgErr CINRun(int32 *var1, int32 *var2, int32 *var3, int32 *var4)

{
    if (*var2 !=1)              /* direction for motor[3] */
      {dir3 = 'f';}             /* forward 'f' is up; reverse 'r' is down */
    if (*var2 !=2)
      {dir3 = 'r';}

    if (*var4 !=1)              /* direction for motor[4] */
      {dir4 = 'f';}             /* forward 'f' is left; reverse 'r' is right */
    if (*var4 !=2)
      {dir4 = 'r';}

    motor = 34;                 /* initialize ports for motors 3&4 */
    md2_init( );
    move_action = 's';
    speed[3] = 14000;
    steps_to_move[3] = *var1;
    direction[3] = dir3;
    md2_move( );
    speed[4] = 14000;
    steps_to_move[4] = *var3;
    direction[4] = dir4;
    md2_move( );
    md2_reset( );
    return noErr;
}

CIN MgErr CINInit(void) {return noErr;}
CIN MgErr CINDispose(void) {return noErr;}
CIN MgErr CINAbort(void) {return noErr;}
CIN MgErr CINLoad(RsrcFile rf) {return noErr;}
CIN MgErr CINUnload(void) {return noErr;}
CIN MgErr CINSave(RsrcFile rf) {return noErr;}
```

APPENDIX B. COLLECT.VI GRAPHICAL CODE

This appendix contains all graphical code for COLLECT.VI. Each block diagram executes in the order shown (exactly as it occurs in the program).

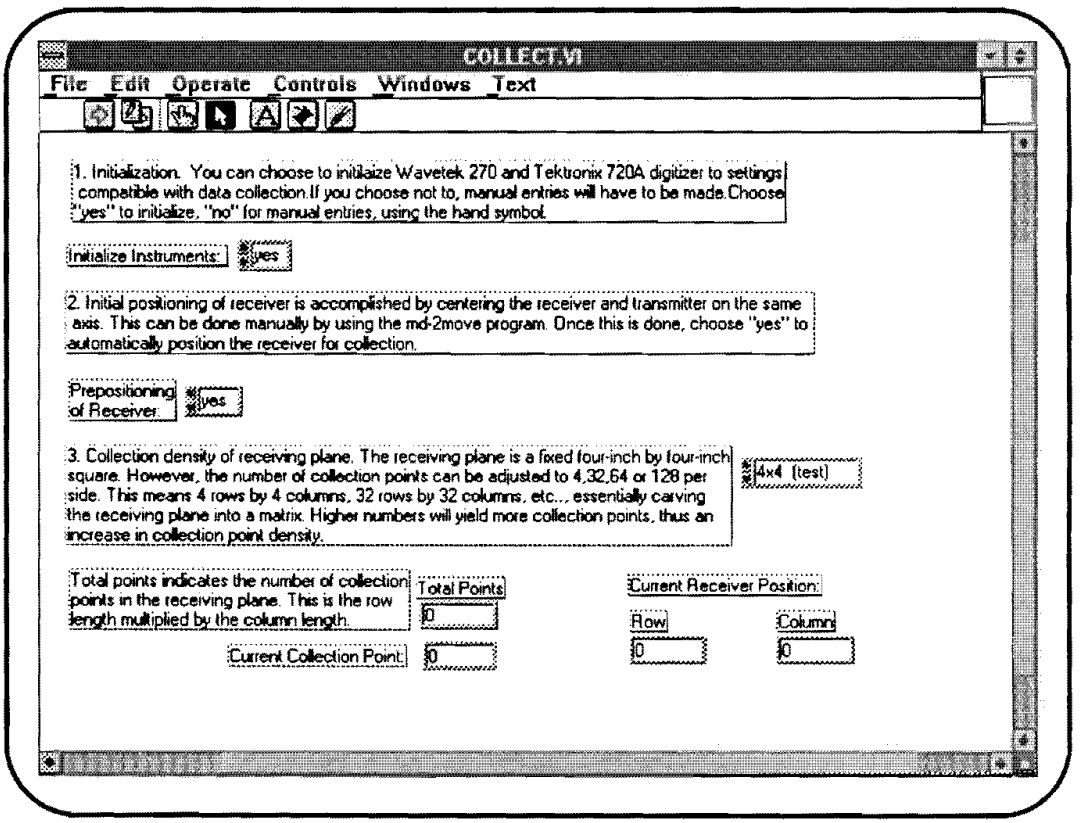

Figure 3. Front Panel.

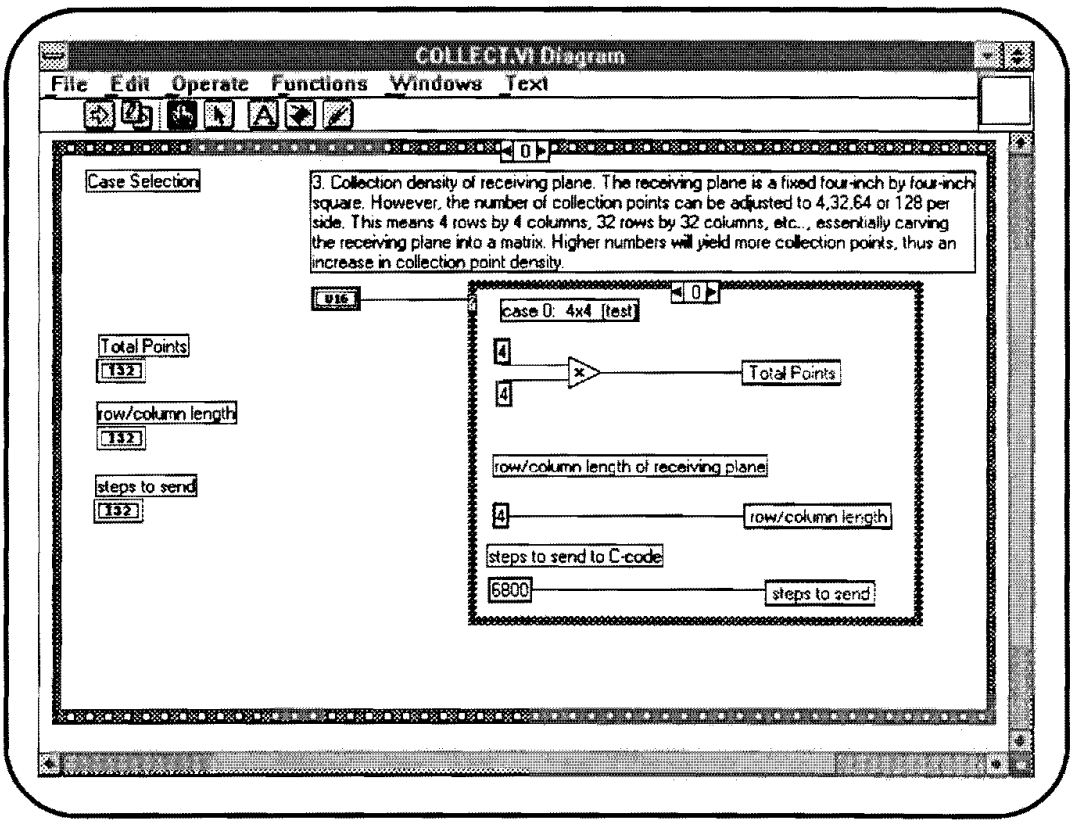

Figure 4. Block Diagram 00. 4x4 Collection Density Setup.

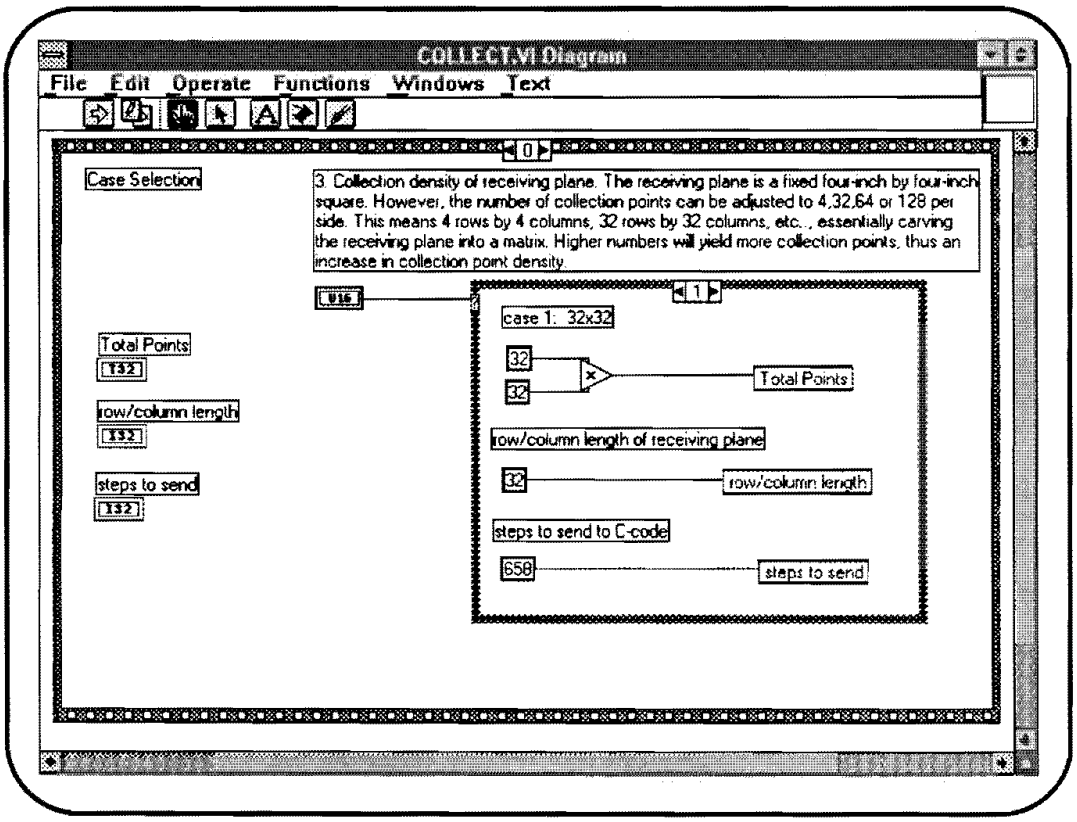

Figure 5. Block Diagram 01. 32x32 Collection Density Setup.

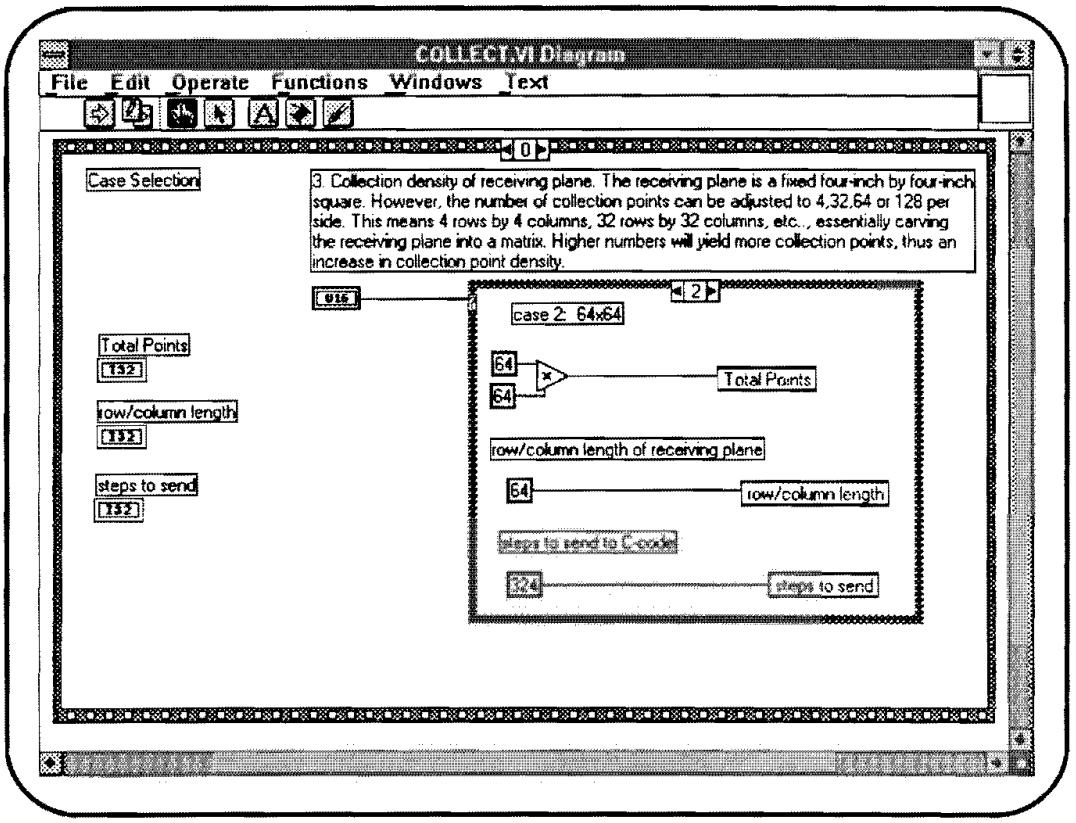

Figure 6. Block Diagram 02. 64x64 Collection Density Setup.

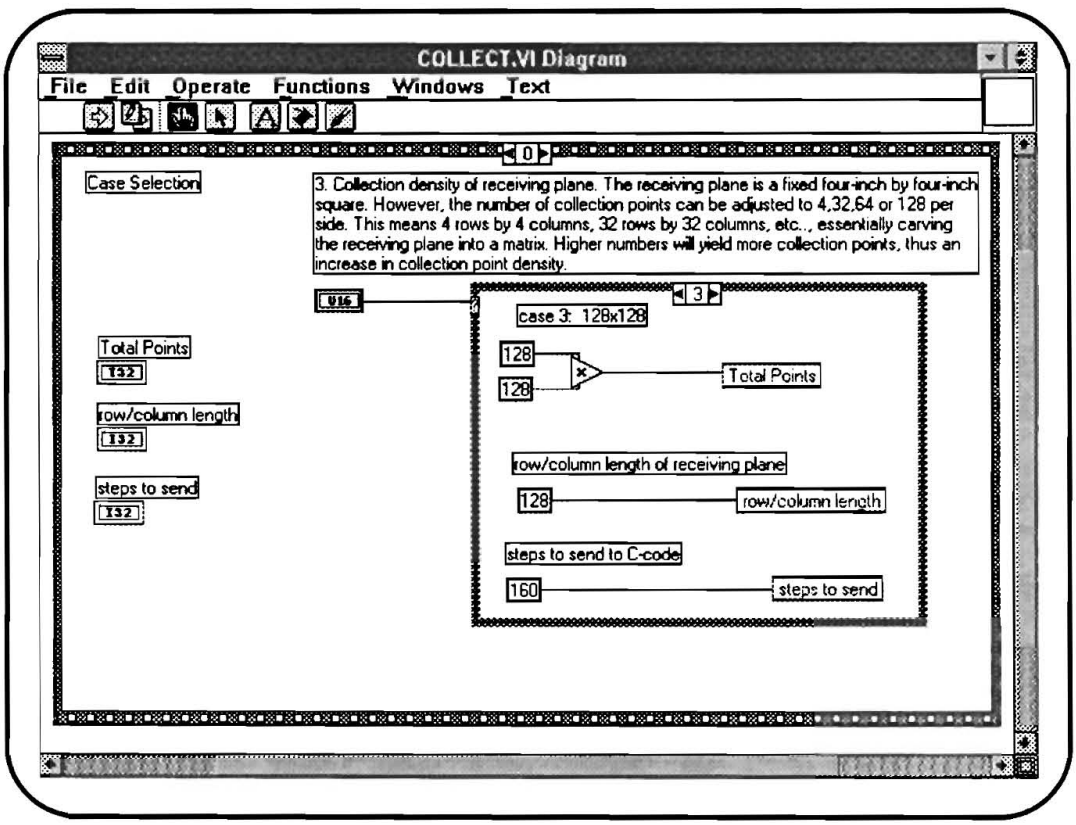

Figure 7. Block Diagram 03. 128x128 Collection Density Setup.

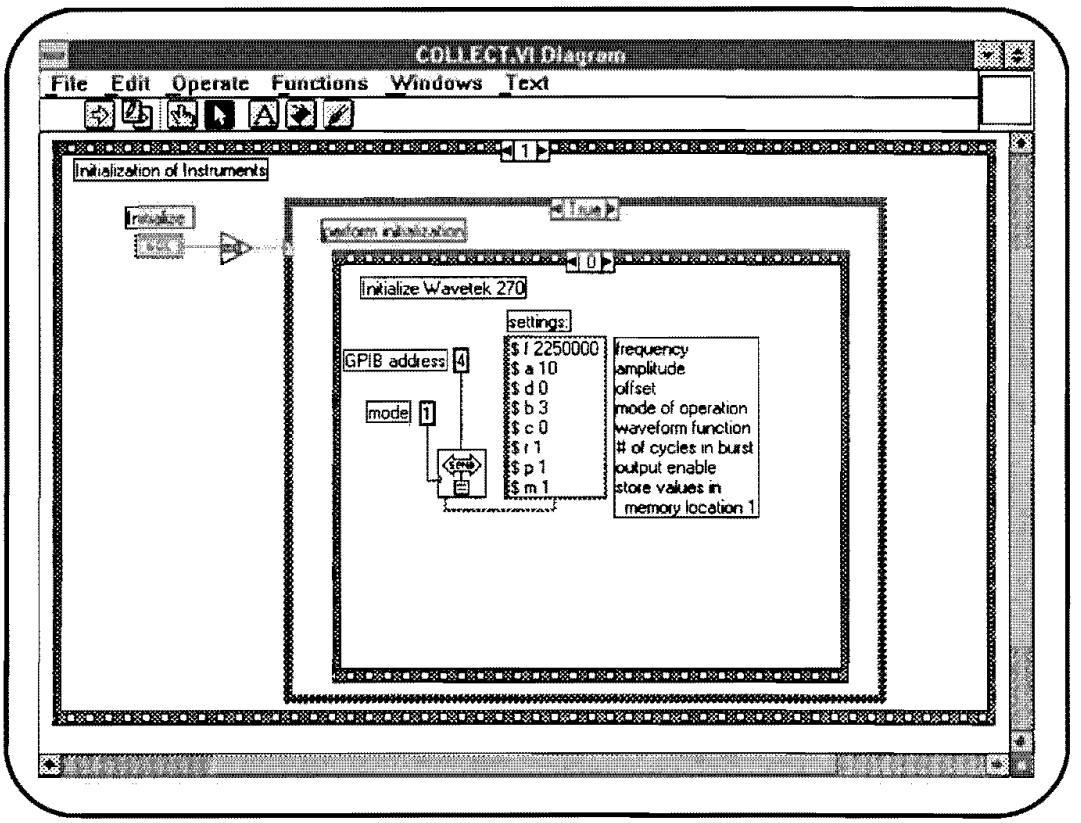

Figure 9. Block Diagram 1T0. Initialization of Wavetek 270.

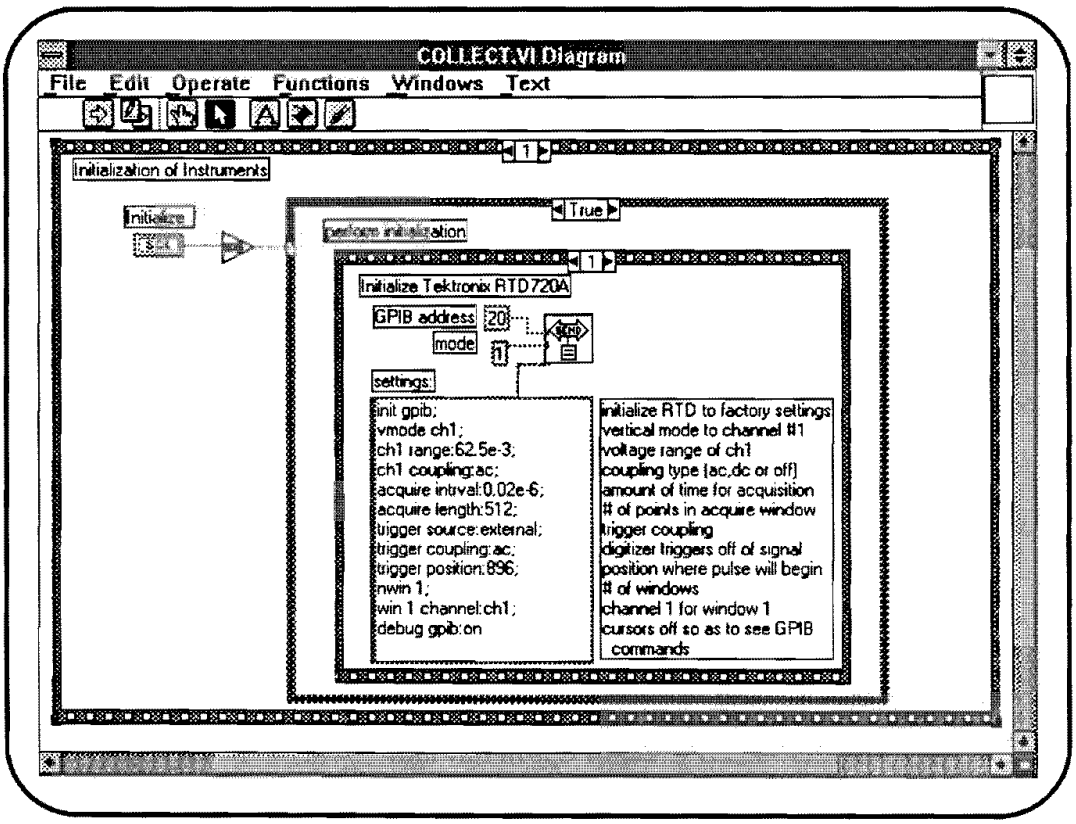

Figure 10. Block Diagram 1T1. Initialization of Tektronix RTD720A.

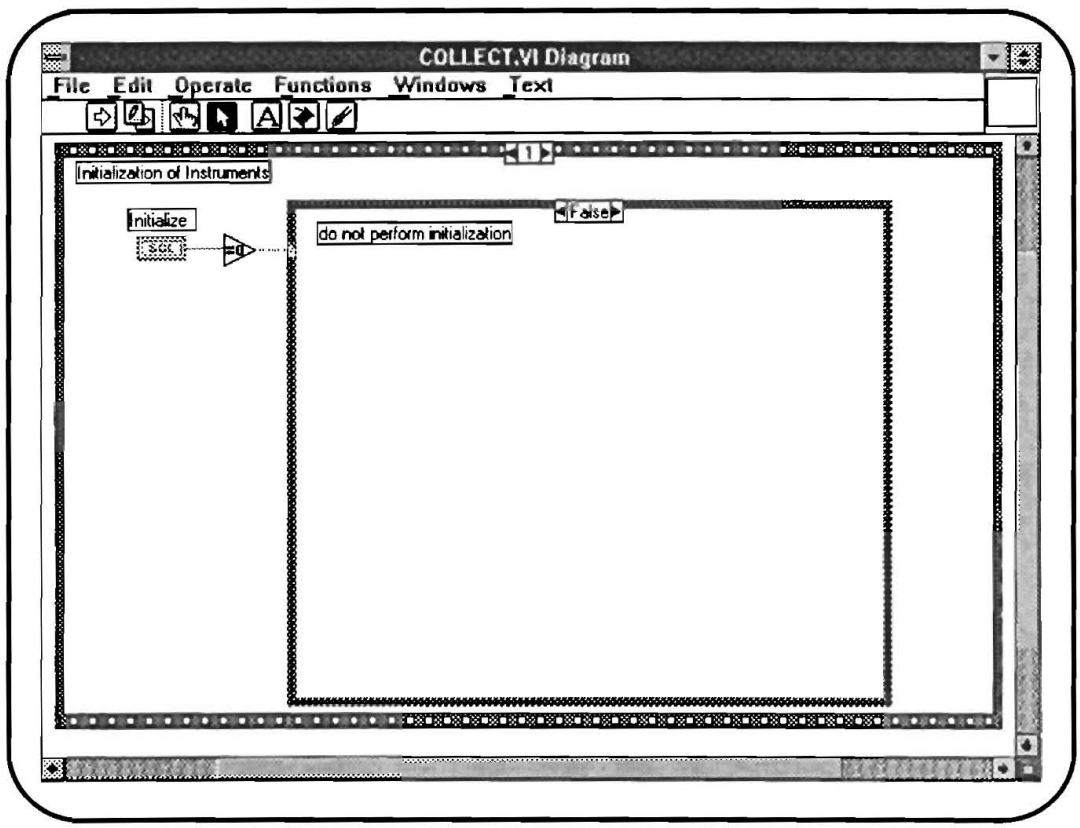

Figure 11. Block Diagram 1F. Initialization Bypass.

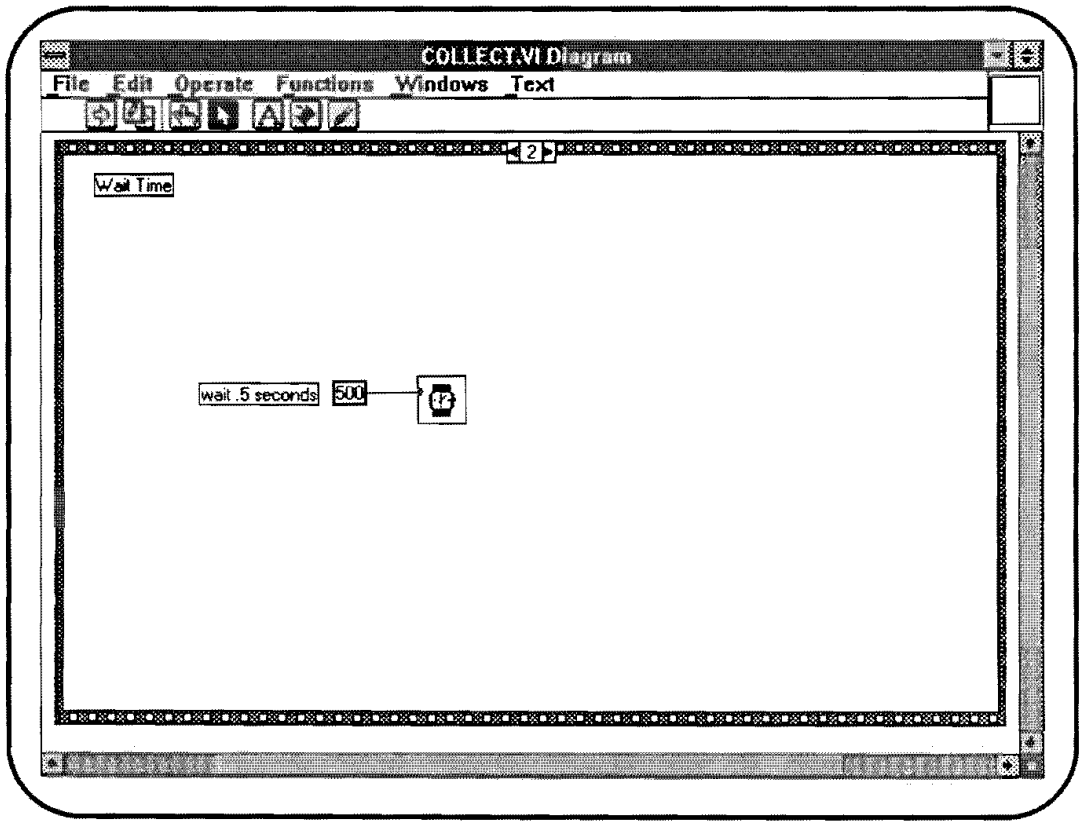

Figure 12. Block Diagram 2. Wait Time of 0.5 Seconds.

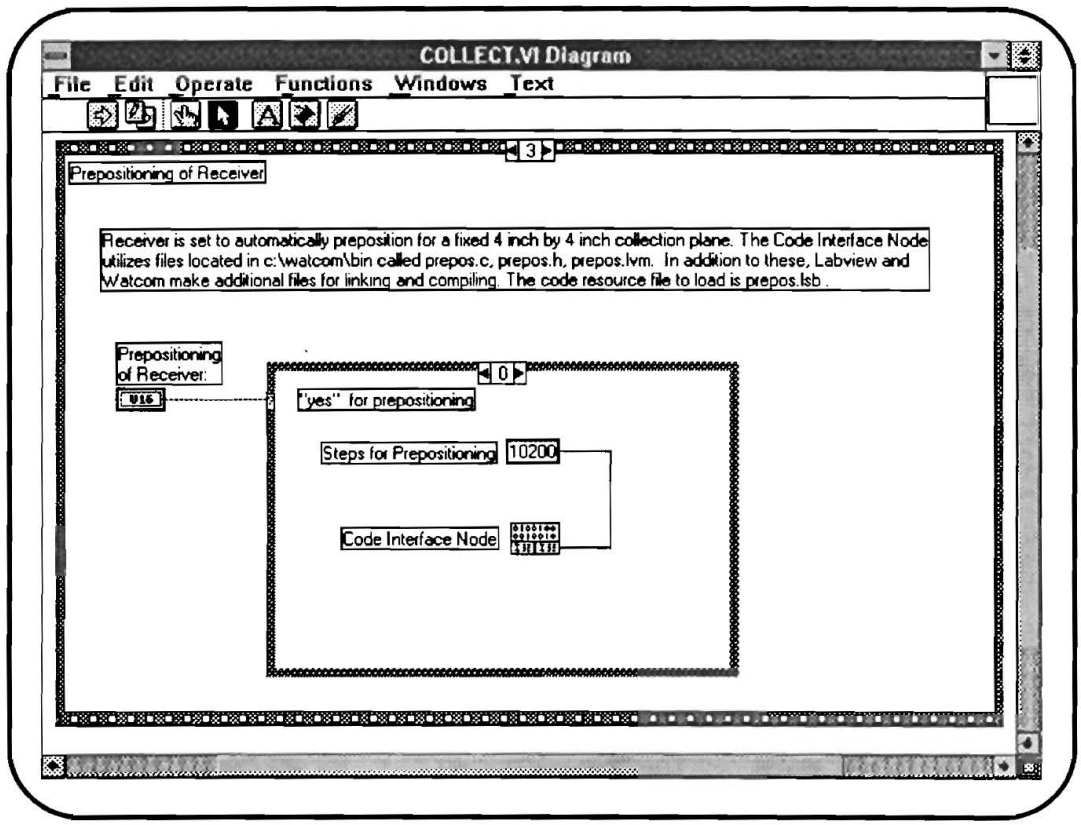

Figure 13. Block Diagram 30. Prepositioning of Receiver.

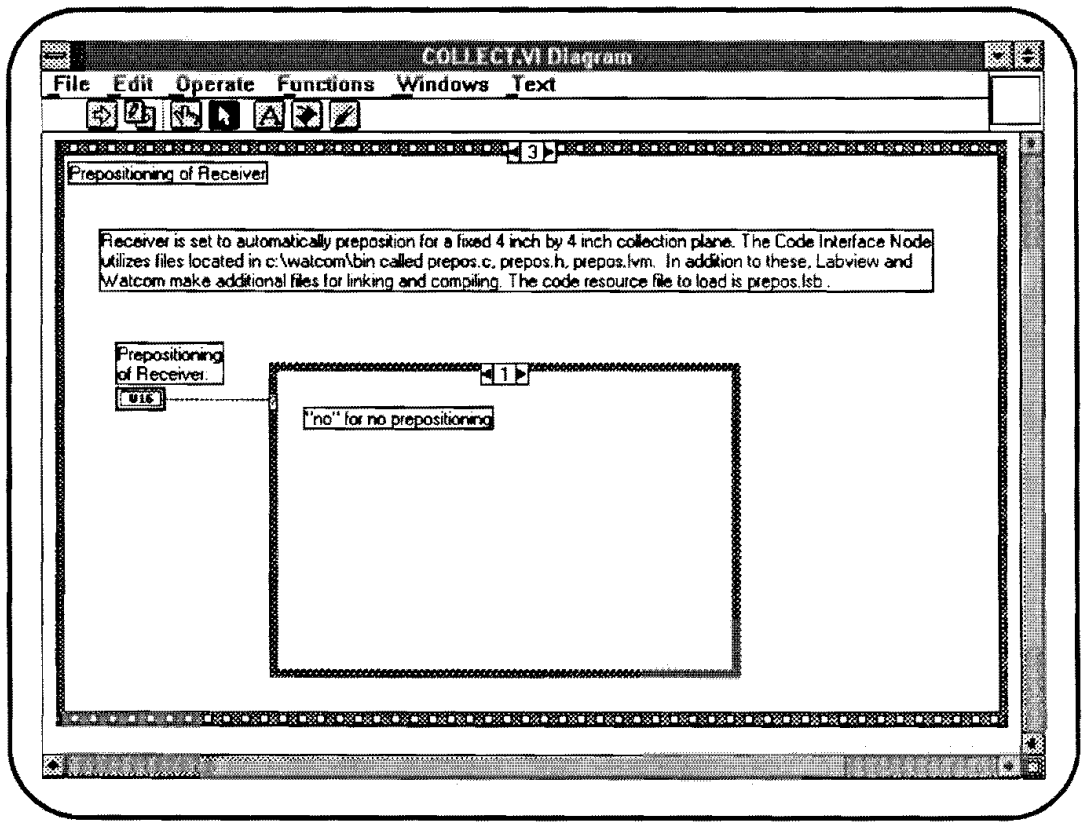

Figure 14. Block Diagram 31. Bypass Prepositioning.

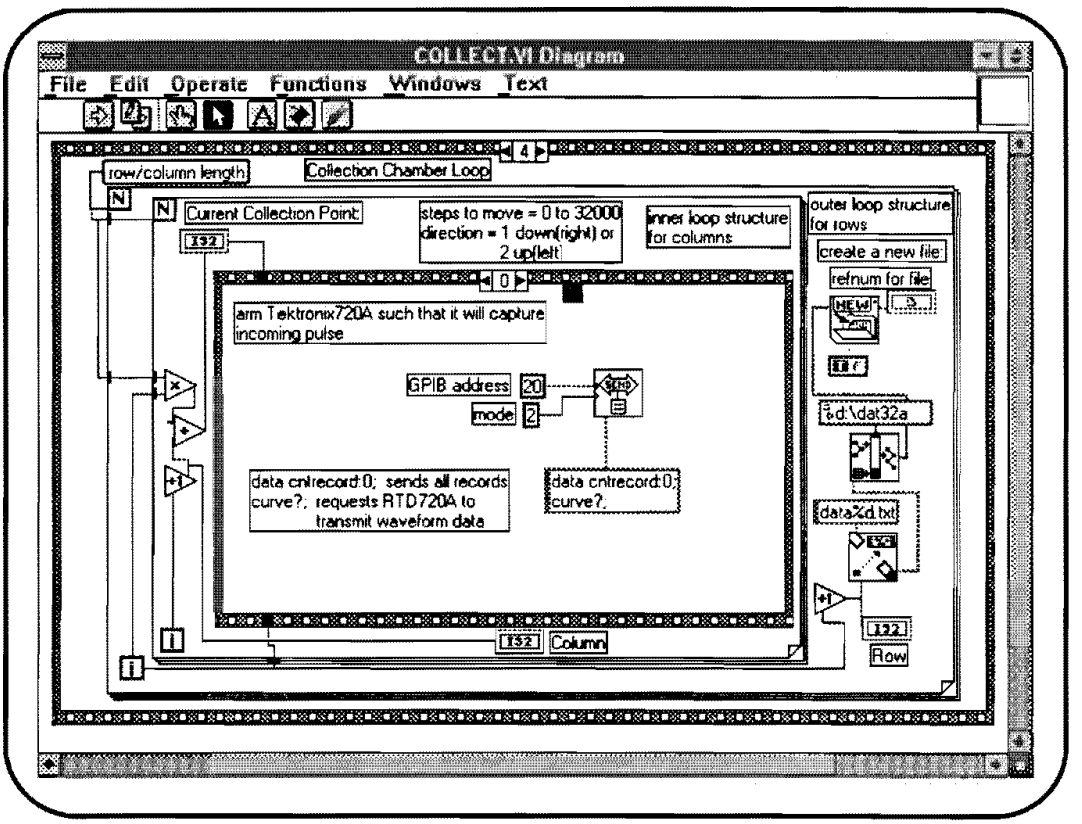

Figure 15. Block Diagram 40. Arming of the Tektronix RTD720A.

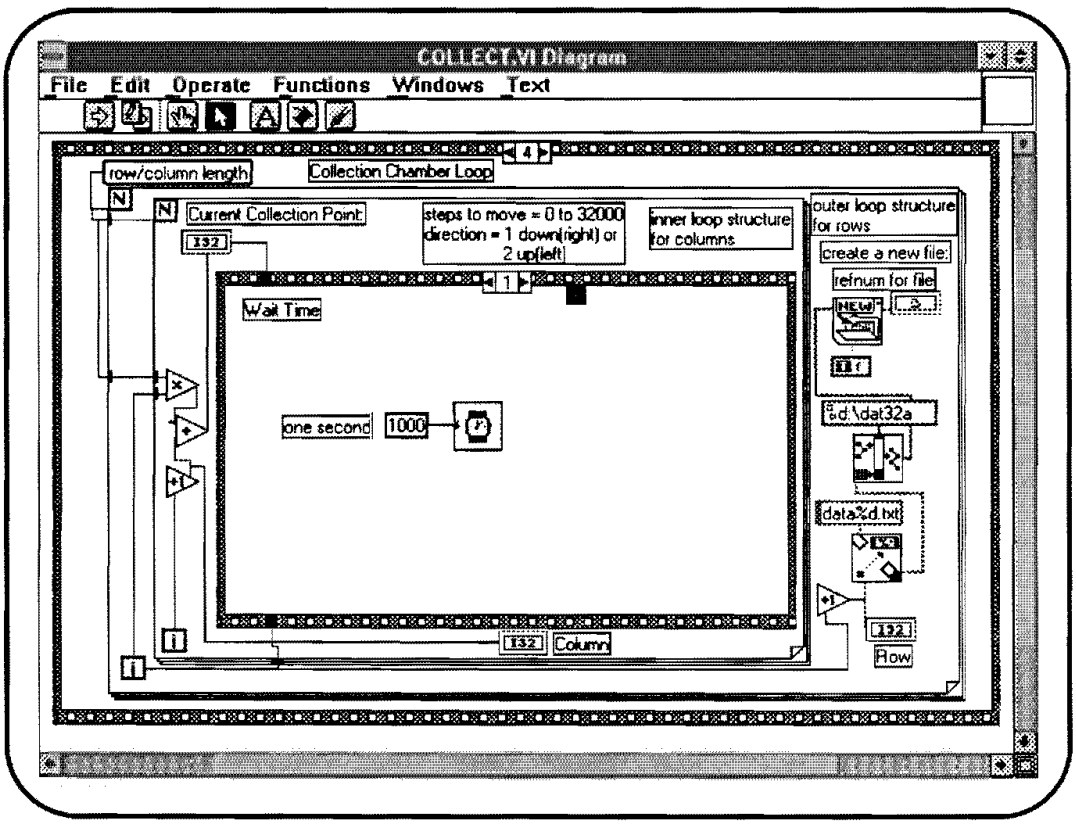

Figure 16. Block Diagram 41. Wait Time of 1.0 Seconds.

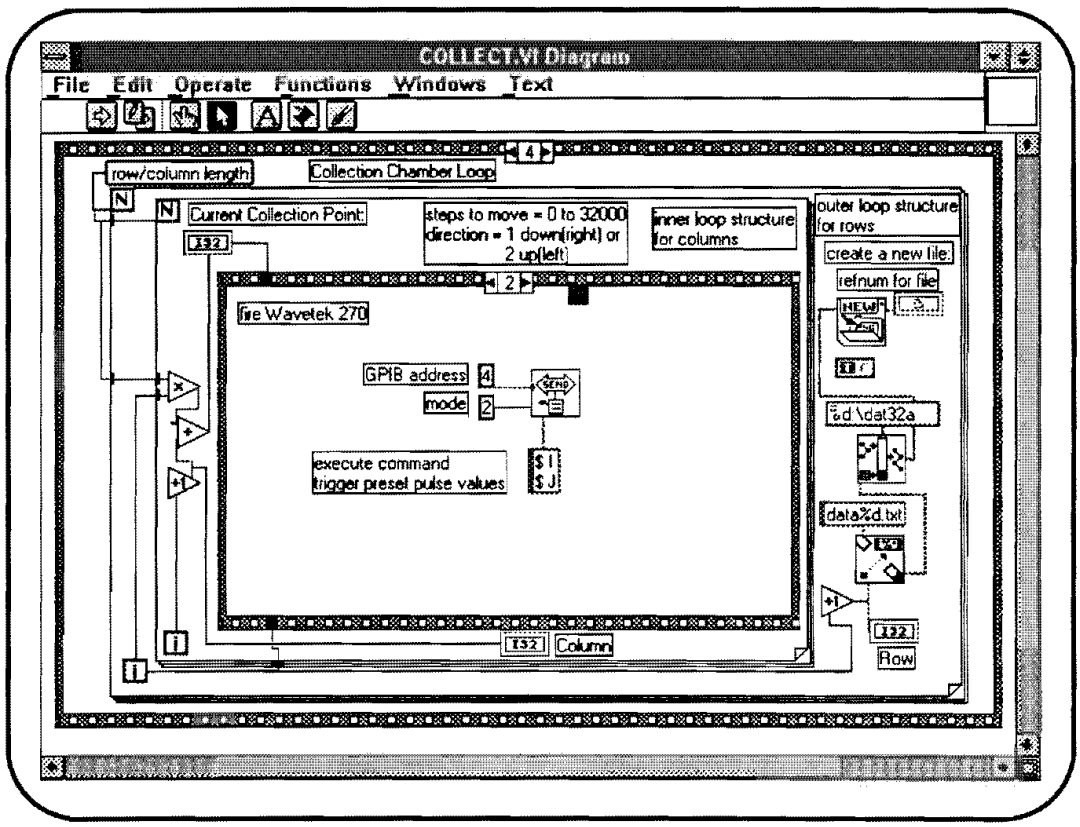

Figure 17. Block Diagram 42. Wavetek 270 Pulse Firing.

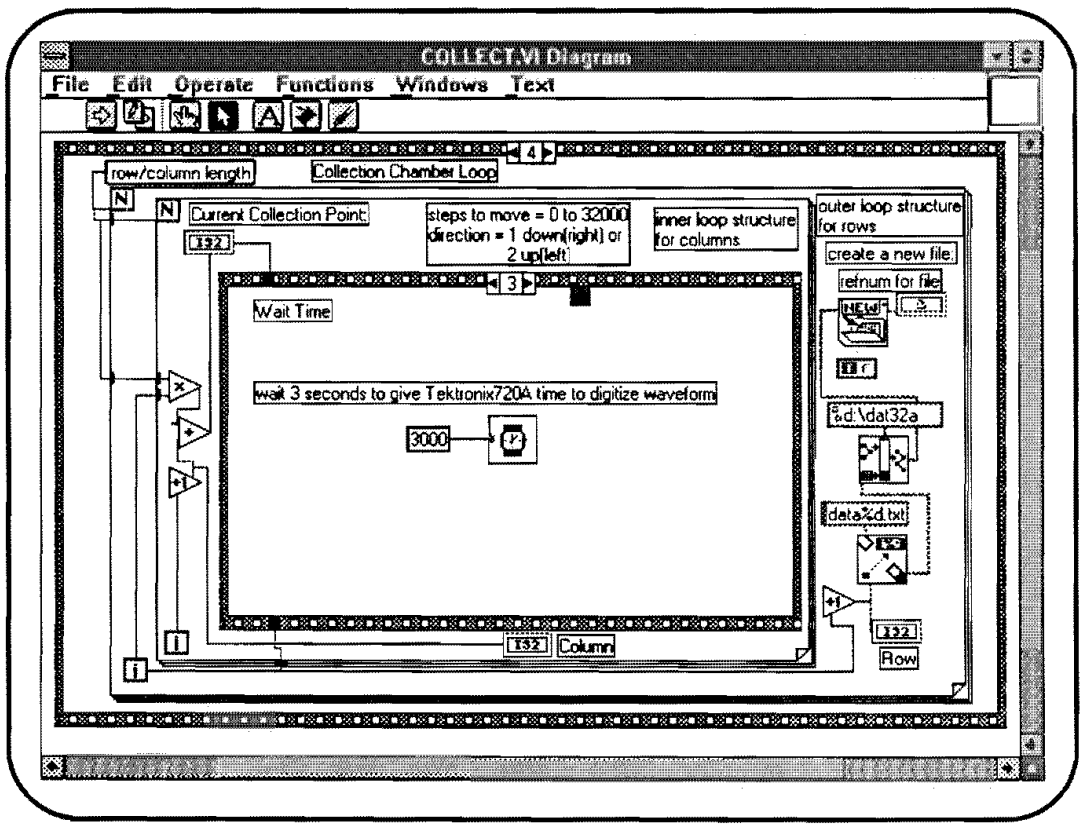

Figure 18. Block Diagram 43. Wait Time of 3.0 Seconds.

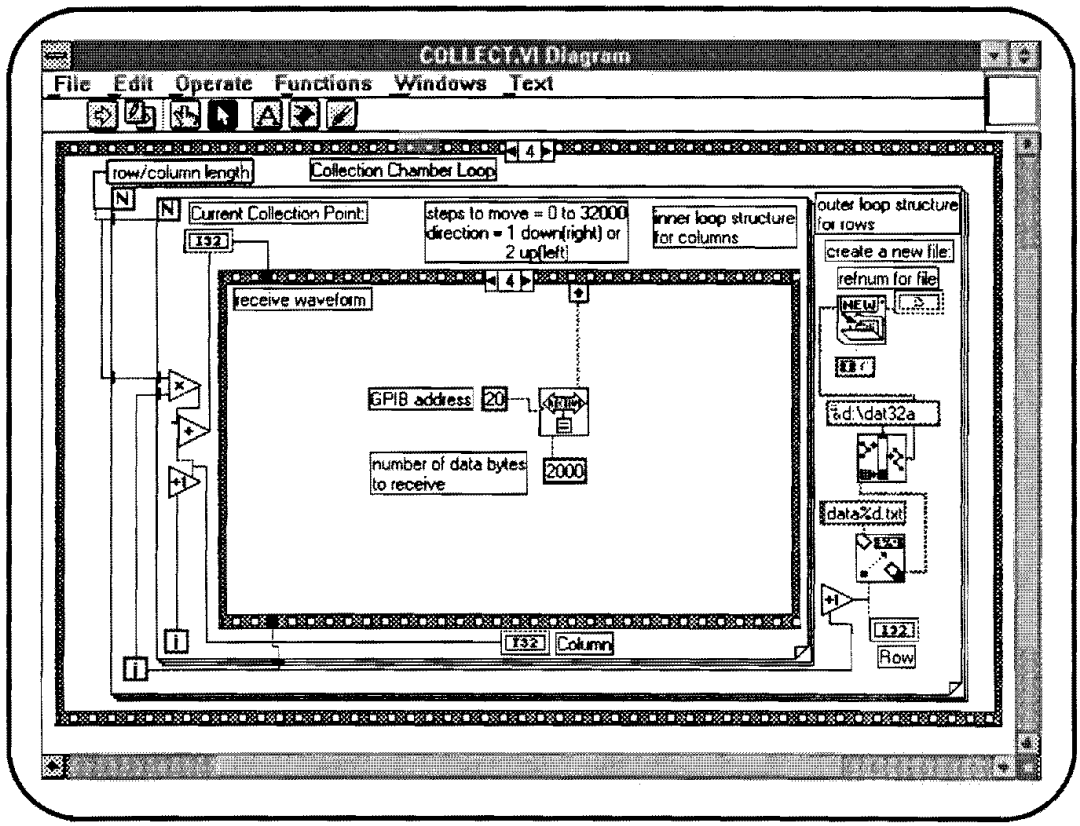

Figure 19. Block Diagram 44. Waveform Reception from Digitizer.

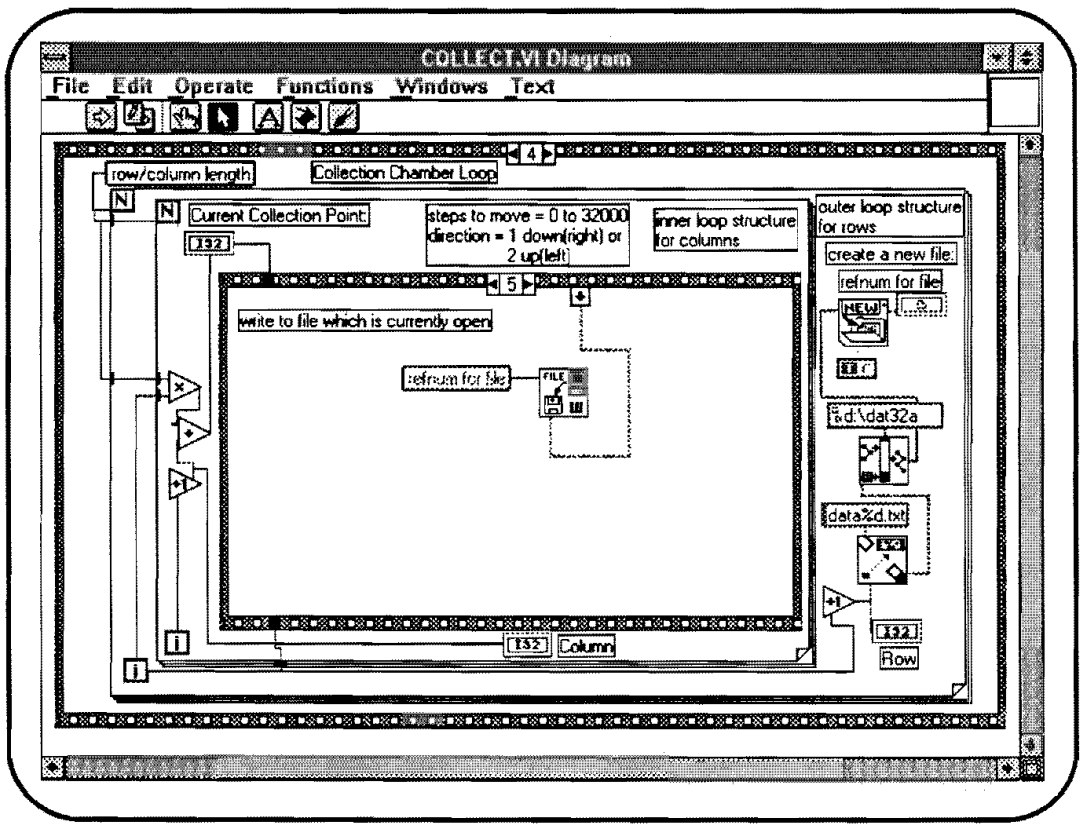

Figure 20. Block Diagram 45. Write to Data File.

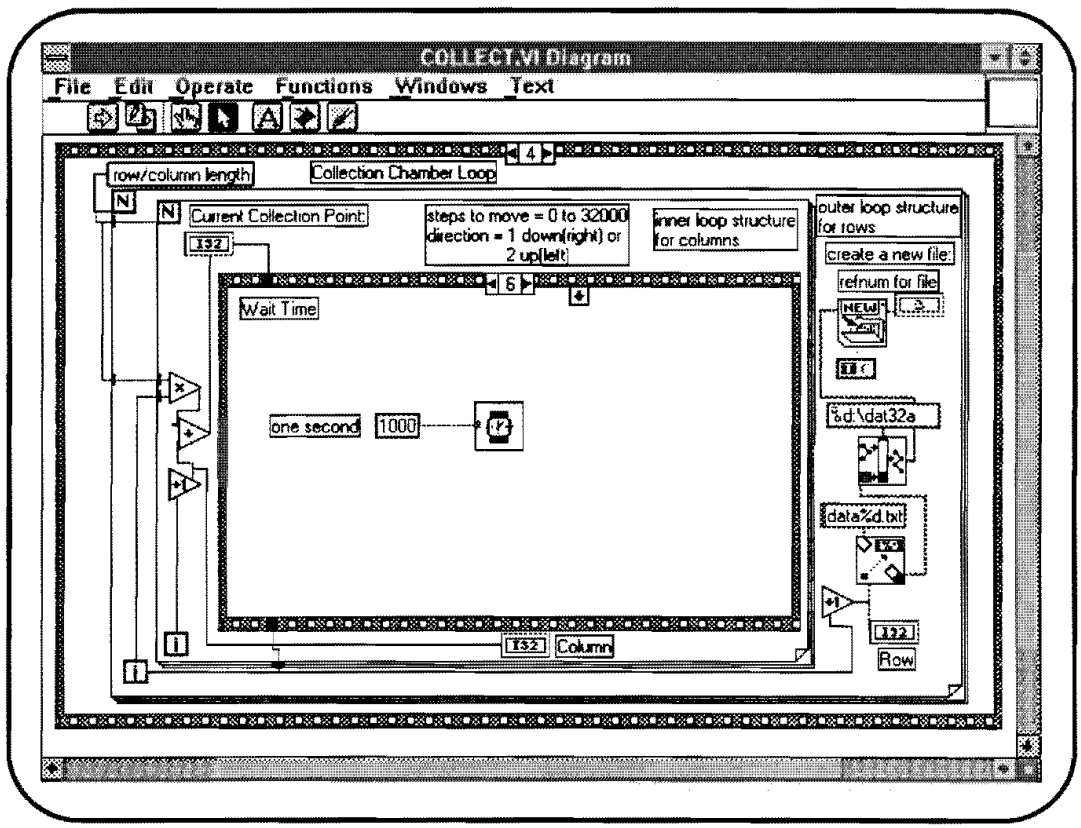

Figure 21. Block Diagram 46. Wait Time of 1.0 Seconds.

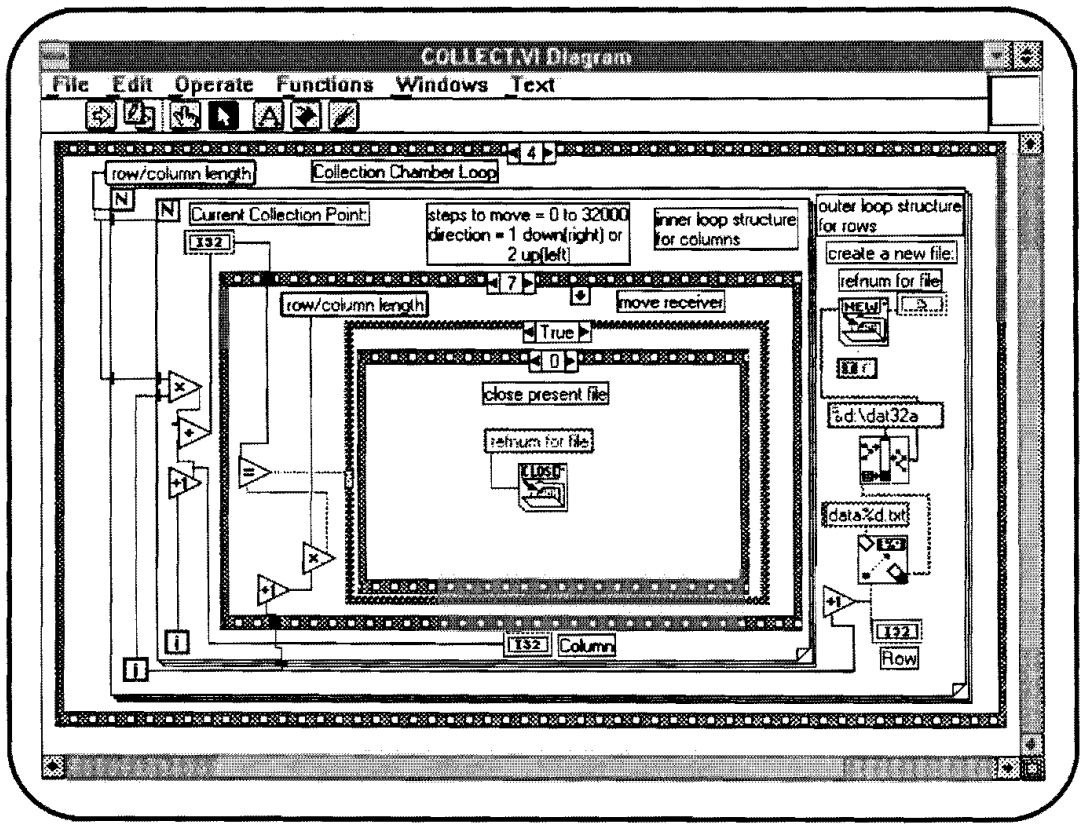

Figure 22. Block Diagram 47T0. Close Current File.

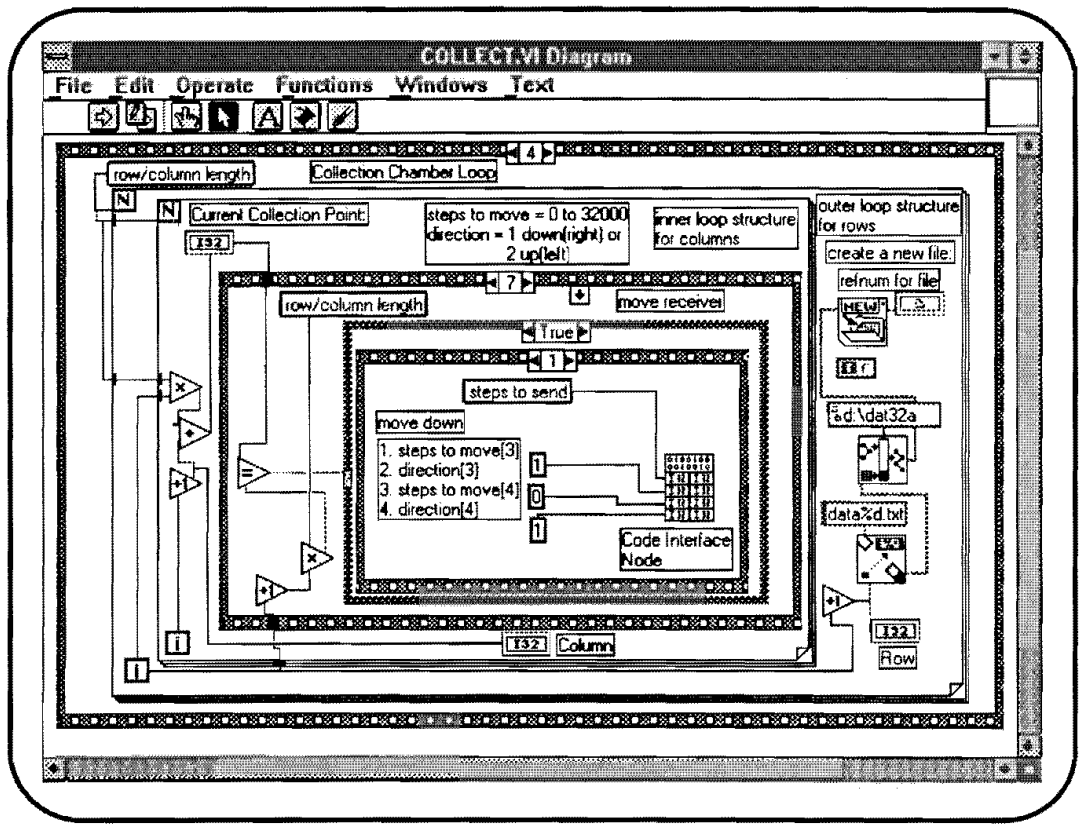

Figure 23. Block Diagram 47T1. Move Receiver Down.

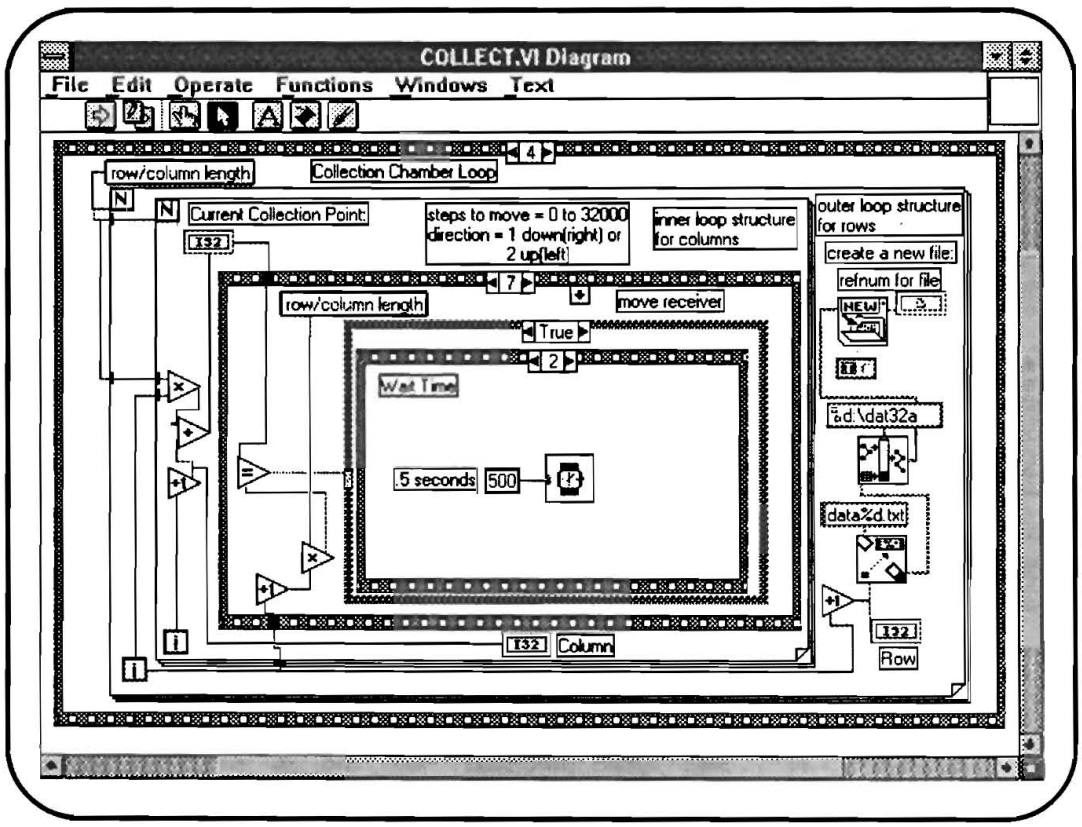

Figure 24. Block Diagram 47T2. Wait Time of 0.5 Seconds.

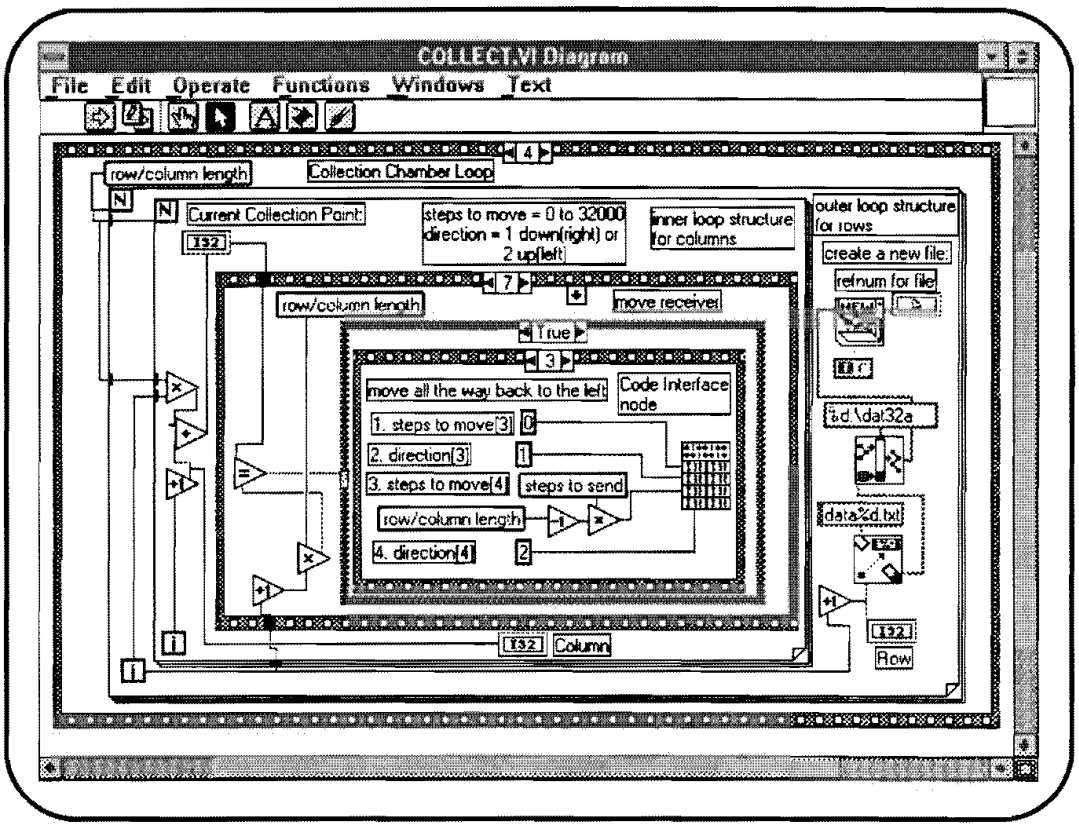

Figure 25. Block Diagram 47T3. Move Receiver Left.

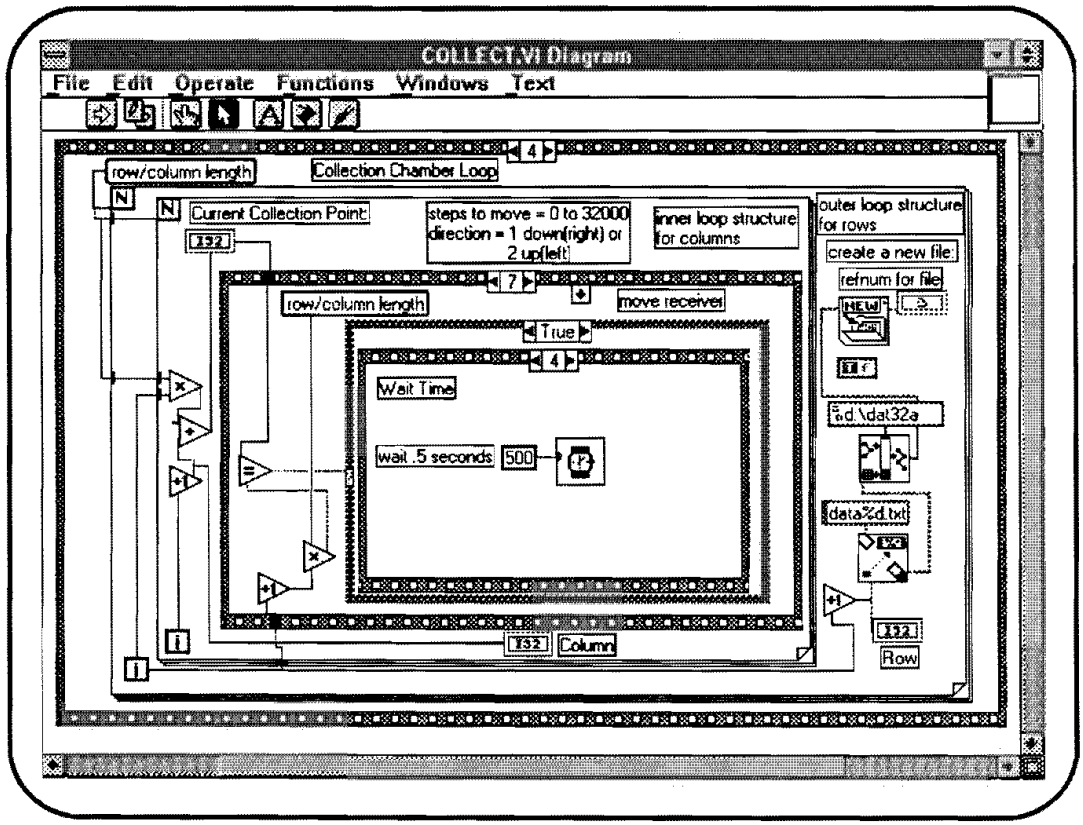

Figure 26. Block Diagram 47T4. Wait Time of 0.5 Seconds.

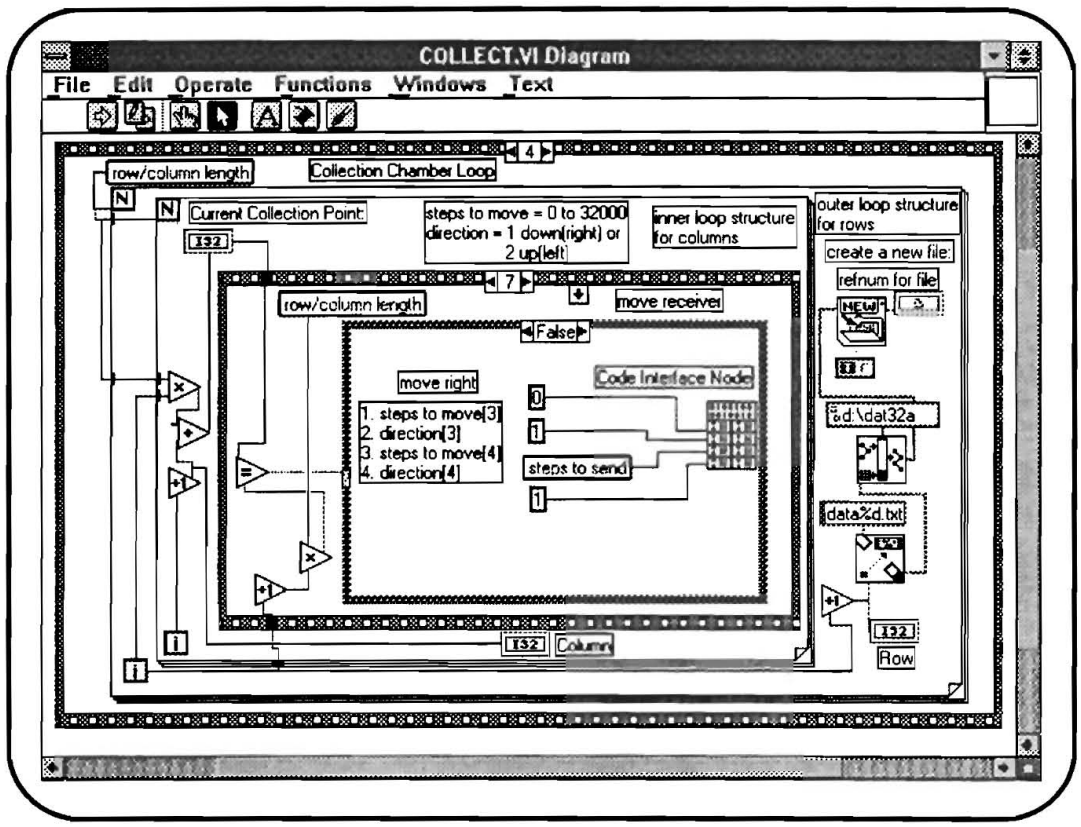

Figure 27. Block Diagram 47F. Move Receiver Right.

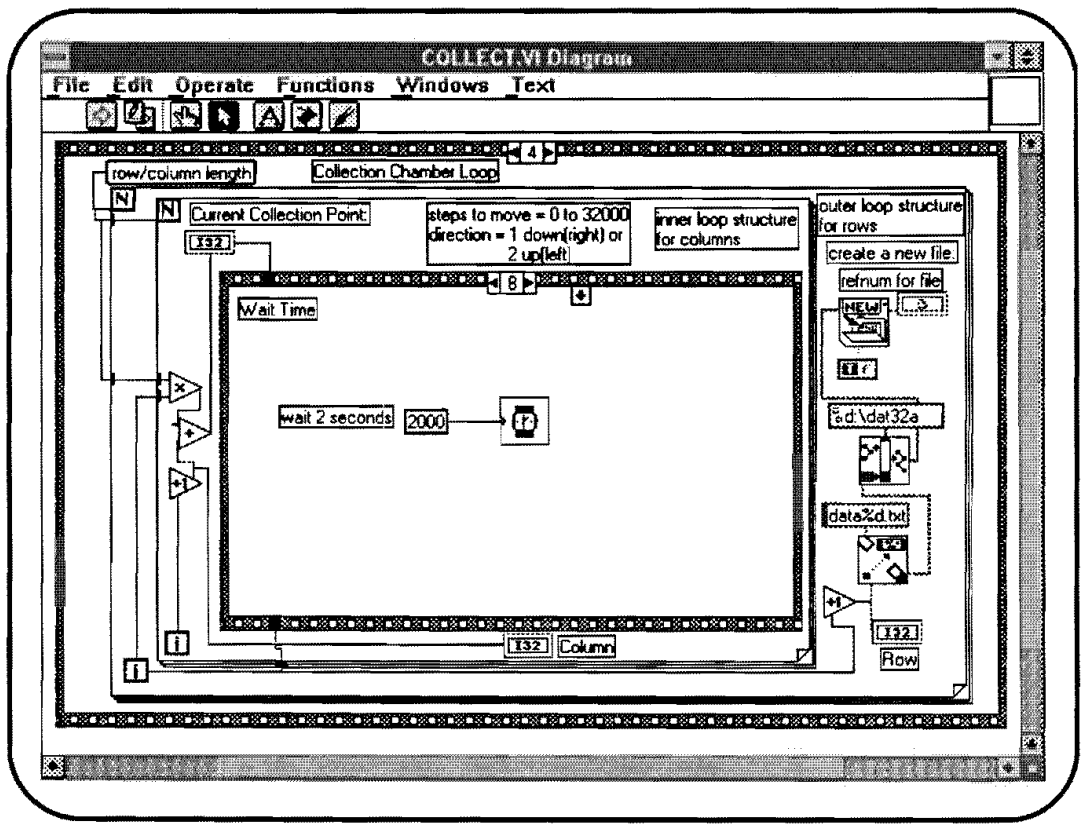

Figure 28. Block Diagram 48. Wait Time of 2.0 Seconds.

LIST OF REFERENCES

1. Sondhi, M.M., "Reconstruction of Objects from their Sound-Diffraction Patterns", *The Journal of the Acoustical Society of America*, Vol. 46, Number 5 (Part 2), pp. 1158-1164, 1969.

2. Goodman, J.W., *Introduction to Fourier Optics*, pp. 48-54, McGraw-Hill Inc., 1968.

3. Reid, W. H., *Microcomputer Simulation of a Fourier Approach to Ultrasonic Wave Propagation*, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1992.

4. Ziskin, M.C. and Lewin, P.A., *Ultrasonic Exposimetry*, CRC Press, Inc., 1993.

5. ARRICK Robotics, *MD-2 Dual Stepper Motor System User's Guide*, Revision B, Hurst, Texas, 1991.

6. Velmex, Inc., *Unislide Motor Driven Assemblies*, Bloomfield, New York, Catalog M-73.

7. Wavetek San Diego, Inc., *Preliminary Instruction Manual Model 270 12 MHz Programmable Function Generator*, 1982.

8. Strum, R.D., and Kirk, D.E., *First Principles of Discrete Systems and Digital Signal Processing*, pp. 54-56, Addison-Wesley Publishing Company, 1989.

9. Tektronix, Inc., RTD720A *Transient Digitizer User Manual*, 1992.

10. Kikusui International, Contract No. F41608-82-D-0337, *Oscilloscope 100 MHz Technical Manual*, MFG P/N COS6100M, July 1985.

11. Iomega Corportion, *The Bernoulli Box II Removable Cartridge Drive User's Manual and Reference Guide*, 1989.

12. National Instruments, *NI-488.2 Software Reference Manual for MS-DOS*, March 1992 Edition.

13. WATCOM International Corporation, *WATCOM C/C++ 32 User's Guide*, 1st Edition, 1993.

14. National Instruments, *LabVIEW for Windows*, Austin, Texas, August 1993 Edition.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3. Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5002	1
4. Professor John P. Powers, Code EC/Po Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5002	4
5. Professor Ron J. Pieper, Code EC/Pr Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5002	1
6. Lt. Peter A. Gatchell, USNR 2 University Circle. Naval Postgraduate School Box 2112 Monterey, California 93943	1